2

Optical Symbolic Processor for Expert System Execution

Final Technical Report

Period of Performance: June 1, 1986 to December 31, 1989

Sponsored by:

Air Force Office of Scientific Research

and

Advanced Research Projects Agency (DoD)
ARPA Order No. 5794
Contract #F49620-86-C-0082

Prepared by:

Julian Bristow
Principal Research Scientist
Honeywell Systems and Research Center
Bloomington, Minnesota

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

REPORT DOCUMENTATA ION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE

3. REPORT TYPE AND DATES COVERED
FINAL 01 Jun 86 TO 31 Dec 89

4. TITLE AND SUBTITLE
Optical Symbolic Processor for Expert System Execution

5. FUNDING NUMBERS
ARPA ORDER NO 5794

6. AUTHOR(S)
Dr Anis Husain

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Honeywell Inc
Corporate Technology Center
10701 Lyndale Ave
Bloomington MN 55420

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 90 . 0744

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NE
Bldg 410
Bolling AFB Washington DC 20332-6448
Dr. Alan E. Craig

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
F49620-86-C-0082

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal of this program was to develop key concepts for optical symbolic computing. During the course of the program, both a top-down and bottom-up approach was taken to develop an architecture for symbolic computing. The approach was intended to result in an architecture suitable for the design goals while being implementable with practical components.

Key results of the program include the following:

- Design of a unique symbolic processing architecture,
- Identification of lack of suitable addressable optical memory as a major impediment in the implementation of existing paradigms,
- Demonstration of a unique, critically needed polarization based modulator with 17 dB extinction ratio, and demonstration of the world's highest extinction ratio (23 dB) of AlGaAs/GaAs modulator at 1 GHz,
- Demonstration of high density modulator arrays with the world's smallest pitch (20 um) and less than 20 dB crosstalk,

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C -	Contract	PR -	Project
G -	Grant	TA -	Task
PE -	Program Element	WU -	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA -	Leave blank.
NTIS -	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

OPTICAL SYMBOLIC PROCESSOR FOR EXPERT SYSTEM EXECUTION

FINAL TECHNICAL REPORT

Period of Performance: June 1, 1986 to December 31 1989

Sponsored by:

Air Force Office of Scientific Research

and

Advanced Research Projects Agency (DOD)
ARPA Order No. 5794
Contract #F49620-86-C-0082

Prepared by:

Julian Bristow

Principal Research Scientist
Honeywell Systems and Research Center, Bloomington, Minnesota

Prepared by: ______________ Date: 5/11/90
Julian Bristow

Approved by: AHusain Date: 5/11/90
Anis Husain
Section Head

Approved by: AHusain for D. Lamb Date: 5/11/90
- David Lamb
Department Head

The views and conclusions contained in this document are those cf the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Contents

1. Summary

The goal of this program was to develop key concepts for optical symbolic computing. During the course of the program, both a top-down and bottom-up approach was taken to develop an architecture for symbolic computing. The approach was intended to result in an architecture suitable for the design goals while being implementable with practical components.

Key results of the program include the following:

- Design of a unique symbolic processing architecture

- Identification of lack of suitable addressable optical memory as a major impediment in the implementation of existing paradigms

- Identification of the interconnection network as the major performance limitation for massively parallel systems

- Demonstration of a unique, critically needed polarization based modulator with 17 dB extinction ratio, and demonstration of the world's highest extinction ratio (23 dB) of AlGaAs/GaAs modulator at 1 GHz

- Demonstration of high density modulator arrays with the world's smallest pitch (20 μm) and less than 20 dB crosstalk

The approach initially taken was to examine computational models of computer languages, determine primitive operations required, and develop and evaluate a conceptual architecture. It was found that the computational requirements of logic languages and functional languages are primitive operations which involve manipulation of complex structures such as graphs and trees, and that the execution of the languages can be described as manipulations of those data structures. The representation of the complex data structures imply that the representations must be exact (digital), and that some means to denote connections between the data items, such as pointers, is required. Since the representation between data items is more important than the actual items stored, the most important functions are the manipulations of the data structures.

Examinations of the optical architectures available to represent and implement the identified functions showed that some way to perform location addressable memory was needed. One technique, matrix representation, was identified and a technique to construct addressable optical memories was invented. By examination of a possible architecture however, it was found that these methods do not adequately perform the computational primitives. Moreover, it was found that while functional languages and logic languages require similar primitive operations, implementation of logic languages in parallel optical environments is more difficult.

A technique for digital optical computing, known as symbolic substitution and several variants on this technique were evaluated. It was found that symbolic substitution by itself cannot perform the required memory and data movement functions, but is well suited to the control functions required in a computer. We examined the possibility of combining other optical computing structures with symbolic substitution to perform the data movement and storage to develop a viable computer architecture.

A novel architecture for an optical symbolic computer was developed. The architecture, Symbolic Processing Architecture in Optics, is designed for executing functional language programs using combinator graph reduction. Sparo is designed with the goal of exploiting the available fine-grained parallelism of both combinator graph reduction and primitive optical operations. A planar array of processors communicating by messages over a network provides the processing power of SPARO. The finite state machine of individual processors is expected to be implemented using symbolic substitution techniques, while gatable interconnects would be used for realizing data movements between the processor and the network. We proposed a simple register based network that would enable multiple messages to be delivered concurrently. It is shown that the architecture can easily be scaled to accommodate large combinator graphs. The detailed control sequence required for processing within the nodes and for messages are shown as macro instructions that would be executed by each processor in SPARO. These macro instructions can be further translated into simpler symbolic substitution rules.

Our implementation effort began with a detailed performance evaluation of our optical architecture, SPARO, for combinator graph reduction. Since we had determined that the interconnection network was the bottleneck in the performance of the architecture, our focus was on the message throughput of the simple register-based network. We derived an accurate performance model for the equivalent bidirectional ring network and found, both by analysis and simulation, that the net parallelism in the architecture was restricted by the low message traffic in the network. When messages exhibited no locality, the throughput for a 1024 processor network was limited to 8. With local messages, the maximum throughput for the same network was 27.

The poor performance of the simple ring network motivated us to examine other more elaborate but efficient interconnection network topologies. The alternatives considered were hypercubes, multistage interconnection networks (MINs), and single-stage shuffle-exchange networks (SENs) and replicated SENs. On the basis of analysis and extensive simulations, we found SENs, especially replicated SENs, to be the most feasible and promising. Recent investigations have indicated that SENs could be implemented efficiently in optics. Furthermore, we established that replicated SENs can provide a high throughput competitive with any other interconnection network.

While the shuffle connection of the SEN is feasible in optics using passive devices, a full-scale exchange switch which handles conflict resolution among competing messages is much more difficult. The functionalities required for the exchange switch and its controls were therefore analyzed. These functionalities were then assessed for optical implementation. A reasonable approach appeared to be to construct the basic exchange switch, and then incrementally add the necessary functionalities. We found that while the basic switch and the representation of the message can be done with relative ease in optics using different information encoding techniques, the conflict resolution function is far too complex to be implemented optically. Even using brute-force techniques such as holographic look-up tables to implement combinational logic that underlies the exchange switch, a large network (1024 or more) would require exchange switches of prohibitive sizes. We conclude that optically controlled network exchange switches will be a reality only when optics technology promises basic switching logic to be competitive in size and speed with electronics.

In conjunction with our work on the optical exchange switch, we also evaluated the advantages and the relative feasibility of hybrid optical designs for the complete SEN. We evaluated electronic and hybrid SEN implementations in terms of complexity and performance. The hybrid design refers to the use of an electronic exchange switch in conjunction with an optical shuffle connection. The analysis of electronic SENs required designing the interface between the processors and the SEN, the smart exchange switch, and means of laying out the perfect shuffle within the board. We considered both GaAs and ECL technologies to determine the highest performance of an electronic SEN. Our results showed that when a large number (1024 or more) of specialized graph reduction SPARO processors, whose complexity and sizes were estimated on paper, are packed on a board for high speed parallel computing, there is a severe performance degradation due to the limited parallelism in transferring messages. Our focus was therefore directed to using optics for implementing a high-bandwidth and high-density SEN multiboard architectures.

The requirement of high density I/O for boards is not unique to SENs. This was based on our analysis of the general I/O requirements of parallel architectures that are implemented as multiboard systems using other interconnection networks such as crossbars and hypercubes. A formal analysis of board I/O requirements in transferring messages in parallel between PEs was conducted to compare SENs, hypercubes, and crossbars. A particular example, the Connection Machine, was also examined to obtain a real-world reference. It was clear that as larger levels of parallelism are employed, existing electrical connection technology would be hard pressed to provide the high degree of connectivity and parallelism necessary for high performance. Our results revealed that if a large number of boards are used in implementing the architecture, then a single-stage SEN is the best choice, provided that the network load is not very high.

The exchange switch analysis as well as the earlier performance analysis of SPARO motivated us to focus our energies in determining the optical techniques and devices that would be the most promising in providing the high bandwidth and high density interconnections. We therefore compared the properties of both demonstrated and emerging optical interconnection technologies. Rather than merely examine the performance of the interconnection medium, we considered the entire interconnection problem, including the possible implementation of the optoelectronic transducers required to interface with the electronic processing elements. Specific approaches investigated were fibers, polymer waveguides, planar holograms, volume holograms, and bulk optics/microoptics. Our assessment reveals that polymer waveguides offer the most promise if electronic processing elements are to be used in conventional architectures. The choice was driven significantly by the absence of suitable transducers to operate with three-dimensional free-space interconnects, rather than by predictions of attainable interconnection density based on diffraction limits.

A critical element in parallel optical interconnection is the optical source. For systems employing parallel operation of 1024 or more processors, considerations of lifetimes suggest that existing diode lasers will not provide adequate performance. Our favored approach involves employing a small number of high-power lasers in remote locations where their operation may be better controlled. These lasers are then fanned out to high-density modulator arrays. To be immune to the variations of temperature and wavelength likely to encountered in practical; machines, devices relying on the electrooptic effect offer the most promise.

Perusal of the relevant literature reveals that a large number of designs for electrooptic waveguide modulators have been reported. However, the important issue of high-density operation in arrays has not been addressed. This is a consequence of the development of such devices for telecommunications applications, where space is not at a premium. Our effort then focussed on the development of high-density arrays of electrooptic modulators.

A novel waveguide modulator was developed to meet the unique requirements of high density array uses. Constraints of array operation suggest that some means of disposing of the unwanted light should be sought, such that this light does not corrupt the signal in adjacent channels. A possible approach involves the use of polarization rotation modulators and waveguide analyzers. This enables the unwanted light to be converted to heat. Previously reported polarization rotators in waveguide form have suffered from poor conversion efficiency due to imperfect phase matching between the orthogonally polarized eigenmodes. Our design overcomes these deficiencies by employing a separate tuning voltage. Extinction ratios of up to 17dB have been determined for this device.

Arrays of electrooptic waveguide modulators were developed. Both the novel polarization rotator and the more conventional Mach-Zehnder modulators were used to construct arrays with interdevice spacings as small as 20 microns. Arrays of eight working devices were demonstrated. The Mach-Zehnder devices used had the highest extinction ratio reported to date for any III-V modulator, 23dB. The bandwidths of both devices were determined and found to be at least 1GHz. Higher frequency operation may be possible, but could not be verified using the test arrangement.

2. Expert Systems in Optics

One of the most significant advances in the field of artificial intelligence (AI) has been the development of powerful new computer systems known as "expert" or "knowledge based" systems. Expert systems di^fer from other well-known computing systems in that they use a body of domain-specific knowledge, obtained from experts and r^presented explicitly in a special form, to solve problems in that domain. The practicality of these systems can be seen in their increasing presence and proliferation in solving real-world problems in many diverse fields, from agriculture to space technology [1,2]. With such increasing usage of expert systems to solve many hitherto intractable computing problems, the need for a high-performance expert system platform has become imminent. The problem of designing such high-performance and special-purpose architectures is not restricted to expert systems alone but also apply to traditional symbolic processing, of which expert systems are a special class, since knowledge is usually represented using symbolic data. In this report, our use of the term symbolic processing will denote traditional symbolic processing which is dıstinct from other non-traditional computing paradigms such as neural networks.

While special-purpose electronic architectures have been designed and proposed, the potential for using optical techniques for expert system architectures and traditional symbolic processing has received significant attention recently [3-10]. These proposals include tree search engines [8-10], systems for propositional calculus [5,6] and systems for accelerating the execution of logic programs [7]. It is well-known that optical computing techniques and architectures seem well-matched to the tasks of searching and template matching that is required in many expert systems. An example of optics addressing a basic requirement of symbolic processing, searching a large memory for an occurrence of a pattern, is in the use of an optical correlator. In one parallel operation, a correlator can find all instances of a pattern in an image. Unfortunately, as we will show, this example is difficult to exploit for symbolic processing. Although both advantageous and disadvantageous properties of optics in computing have been discussed by several researchers before us, we reexamine them here to uncover the complexities in designing an optical expert system or more generally an optical symbolic processor.

Optical techniques appear attractive for symbolic applications for several reasons. Besides performing matching and correlations over a whole image in parallel, optical systems have the promise to provide very high space- and time-bandwidth systems. Optical systems can be three-dimensional and therefore provide inherent parallelism. Furthermore, unlike in electronics, free space optical signals can propagate through each other with essentially no interaction and crosstalk. However, while these advantages of optics are significant, several limitations must be overcome before optics can compete with electronic computers. These limitations include materials and devices that have small nonlinearities, as compared to electronic ones, and the need for large-scale optical systems to employ very structured interconnects [11-17] to keep the system size tractable. An important limitation that surfaces when considering implementation of certain computing functions is that no digital optical architecture has to date been developed to provide random access or location-based addressable memory. This limitation is found to be significant and is discussed at length in subsequent sections.

Logic and functional symbolic processing languages are representative of those typically used to write expert systems and other AI applications. Among the logic languages our focus has been on a parallel logic language, PARLOG [18], which is a derivative of Prolog and Concurrent Prolog [19]. However, the essential conclusions derived for PARLOG apply equally well to Prolog and many other logic languages. We emphasize PARLOG because, unlike the more popular Prolog, the evaluation of PARLOG is inherently parallel. We consider pure functional languages because computational models for such languages also have the potential for parallel implementation. Languages that fall into this category include pure LISP and SASL [20].

Developing systems which can exploit the advantages of optics for symbolic processing is very challenging for several reasons. Symbolic computing is a field in which no single primary paradigm has emerged but which nonetheless has exhibited considerable growth. We believe this means that, in general, very specialized architectures are not likely to be accepted. In our opinion only architectures which possess some degree of flexibility will be used. In addition to flexibility, there are several other goals which we believe are important for future symbolic processing systems: The system should allow the programmer to exploit as much parallelism as is present in the problem. Furthermore, as desired in the design of electronic parallel computers, the system should not require extraordinary efforts to map a parallel algorithm onto the architecture. Another goal would be to ensure that the serial performance of the system is acceptably high since some symbolic processing problems may inherently have a limited degree of parallelism. Finally, the system should attempt to be similar to present-day computer systems, in as much as there exists a significant base of applications which could make use of expanded capabilities.

In the light of these goals and constraints, our decision to limit the discussion of approaches to traditional symbolic computing is a natural one. This is not to say that the investigation of approaches such as neural networks are not without merit, but rather that the class of problems for which neural networks will be useful are usually outside those solvable by conventional symbolic processing. Our approach necessarily minimizes the likelihood that we will develop a radically different optical architecture and that such an architecture will alter the development of symbolic computing practices. We have decided to accept this limitation of our approach. We believe it unlikely that current symbolic processing theory can suggest a radically different architecture which can both meet the goals previously stated and be uniquely suited to optical implementation.

We have imposed several other criteria. The first was that the architecture should scale to real applications which may have over 10^5 simple facts and rules. This criterion implies that component subsystems that are impractical when scaled, or computing strategies that only work problems of limited size (<100 simple rules), were not pursued.

Another criterion was to develop an architecture with optimum overall performance. While the general aim was to design the optical processing components of the architecture, the final details were to be determined on the basis of theoretical and simulated performance evaluations. This criterion implied the use of optics only where it provided the most benefit. (Unfortunately, these decisions cannot be fully resolved until adequate performance evaluations are conducted.)

Yet another criterion was to attempt to exploit the massive parallelism possible with optical systems and to employ parallel computational models.

Finally, all approaches to provide the primitive computational operations required must take into account the overall task. This global view is necessary so that undesirable interactions between subsystems can be avoided. An example of this would be the need for a large amount of data formatting prior to insertion from one part of the computer to a specialized processor. If this data conversion requires a significant amount of time, then the speed of the subsystem will ultimately be limited by the data conversion operation [21]. Thus, while subsystems can be developed individually, the total system performance must be used to evaluate the utility of the subsystem.

We shall now provide a brief outline of the languages for expert systems. A description of computational models for these languages and the requirements to implement them is then presented. Basic building blocks which can be used to implement these architectures are discussed, and two approaches to developing an optical symbolic computer are expounded. A brief overview of our optical symbolic processing architecture, SPARO (Symbolic Processing Architecture in Optics), is provided. The interested reader will find a more complete description later in this report.

3. Languages for Artificial Intelligence Systems

Typical expert systems, as well as most symbolic processing applications, are written in high-level programming languages. The most popular programming languages for symbolic computing are LISP and Prolog. While it is well-known but not often stated, the semantics of the programming language employed directly influence the efficiency of implementation. There is currently significant research in the area of designing parallel architectures for functional and logic languages using parallel computational models [18,22,23]. It is for this reason that we investigate the languages and computational models underlying symbolic processing.

Languages for implementing expert systems can be divided into three categories: imperative, functional, and logic. We examine each of these in terms of their suitability for parallel implementation.

3.1 Imperative Languages

Imperative languages are characterized by sequences of statements or commands that make incremental changes to a global program state contained in a set of variables. Examples of imperative languages are the traditional programming languages such as FORTRAN, Pascal, and C. Even LISP, in the form in which it is used today, is an imperative language. Because imperative languages incrementally change the global state of the program through variable changes, such as through assignment (i.e., setting the variable to a specific value), they permit uncontrolled side effects. Thus, when one procedure changes the values of variables through assignment, another procedure cannot operate on those variables and thus cannot be scheduled at the same time. The presence of such side effects makes it very difficult to exploit parallelism in such languages by partitioning the problem for independent subcomputations. This does not mean that parallelism is not possible with imperative languages. In certain object-oriented type imperative languages, a current research area, side effects are limited and therefore can be used for parallel implementations. In general, however, the semantics of imperative languages discourage parallelism, and it is difficult to achieve parallelism that is proportional to the amount of data present.

Unlike imperative languages, pure functional languages and logic languages do not use assignment and therefore do not exhibit side effects. The parallelism is implicit in these language classes making them suitable for parallel implementation.

3.2 Logic Languages

Programs in most logic languages are composed of Horn clauses which have the form

head <- body

where head is zero or an atomic formula (a predicate or a relationship with arguments supplied), and body is a conjunction of zero or more atomic formulae. The logical interpretation of a Horn clause is that the body implies the head. An empty Horn clause body is considered true. Therefore, a Horn clause with an empty body states the head is true. Such a Horn clause is called a fact. An empty Horn clause head is considered false. Therefore, a Horn clause with an empty head states that the conjunction of atomic formulas in the clause body is false. The refutation of the body can be used to initiate a resolution-based proof that the body is in fact true. In the course of this proof, all variable combinations that make the body true can be discovered. A Horn clause with no head is therefore called a goal or query. Thus, the query to find the grandfather, X, of Y can be expressed by the Horn clause <parent(X, Z), parent (Z, Y)>.

While the best known logic programming languages is Prolog, its sequential semantics render it inherently unsuitable for parallel processing. In Prolog, the order of the clauses is significant; the "database" (a memory where all relations or formulae are stored) is scanned sequentially from top to bottom when attempting to satisfy a goal. Within a clause, the atomic formulae in the body are satisfied from left to right, in order. Finally, when the search leads to satisfying queries that would not contribute to a solution, the Prolog "cut" operator [24] is executed which prevents the search to be directed back up the chain of satisfied goals. The presence of mechanisms like cut, which introduces side effects, makes Prolog further unsuitable for parallel implementation.

Concurrent logic programming languages, e.g. Concurrent Prolog [19] and PARLOG [18], alleviate the problems posed by the sequential semantics of Prolog. While PARLOG and Concurrent Prolog are very similar, the semantics of PARLOG have been defined to eliminate the runtime management of multiple binding environments, in which the same logic variable is bound to different values when attempting to satisfy different goals [18]. Since these issues amplify the problems with addressable memory to be detailed later, PARLOG was chosen as the logic language to investigate a parallel implementation.

3.3 Functional Languages

Programs in functional languages are essentially definitions and applications of functions, which are used to extract values and not updated variables. There is no notion of operations on named objects, and therefore there are no side effects. Examples of functional languages include pure LISP, FP, and data flow languages such as Id [22]. All are ideal for parallel execution.

4. Computational Models for Symbolic Processing Languages

We now consider the computational models for the logic and functional languages. For reasons stated in the preceding section, we only consider the PARLOG logic language.

4.1 Computational Models for PARLOG

The operational semantics of PARLOG are best understood in terms of the AND/OR process model [18,23]. In this model, a process is created for evaluating a formula and for searching for a candidate clause during the evaluation of a formula. The state of a PARLOG evaluation is represented by a process structure called the AND/OR process tree. The nodes in this tree are processes. The leaf processes are either executable or suspended while waiting for some variable to be bound. The nonleaf processes are not executable. They await results from their child processes. There are two types of nonleaf processes: AND processes and OR processes. A process assumes a type AND if it is to evaluate a conjunction of formulae. A process assumes a type OR if it is to search for a candidate clause among the clauses defining a formula or relation. A PARLOG query is evaluated by first searching for any candidate clause and then evaluating it. During query evaluation, the AND/OR process tree grows and shrinks dynamically.

A parallel abstract machine has been designed for PARLOG [23]. This machine is a multiprocessor consisting of processing elements (PEs) that are connected by an interconnection network. Each PE in turn consists of computing entities that are dedicated to handle message passing between the PEs and also handle the process tree management. The PARLOG data objects (or terms) are represented as Directed Acyclic Graphs (DAGs) in this machine. Data objects and the AND/OR processes are distributed among the various PEs. However, the machine has a single virtual address space. This means that the DAGs and the process tree are linked across PEs. This linkage has important consequences for the optical implementation of PARLOG.

4.2 Computational Models for Functional Languages

In the case of functional languages, there are basically two computational models we need consider: dataflow and reduction. In a dataflow model, the program is compiled into a graph representing the data dependencies. The nodes of such a graph, referred to as operators, represent function applications, while the edges reflect the composition of the functions. The dataflow graph is executed directly; an operator "fires" whenever its input arguments are present, sending any output to its direct descendants.

In graph reduction, the program is viewed as a set of rewrite rules. The left-hand side of each rule corresponds to a function specification, the right-hand side to the function definition. In order to evaluate a function, first a directed graph that captures the rewrite information is constructed. The nodes in this graph correspond to functions. The immediate descendants of a node correspond to the definition of the function. The computation can proceed in either a demand-driven, that is, evaluation proceeds only when requested, or in an eager manner, that is, evaluation may be initiated in anticipatory fashion to increase parallelism in execution. After a function is evaluated, it is replaced by its value, hence the name reduction. Eventually, the whole graph is replaced by one value.

Reduction comes in two varieties: string reduction and graph reduction. In string reduction, every occurrence of a variable is treated as a distinct copy, while in graph reduction, all occurrences share the same copy. A technique called combinator reduction is often used for efficiently executing functional language programs. In this technique, variables occurring in a function definition are abstracted out to produce a function definition consisting solely of operators called combinators. Combinators are higher order functions. That is, they can accept functions as arguments and return functions as results. The so-called S, K, and I combinators [20] are sufficient to remove all variables from any function definition. Combinator graph reduction involves two steps. First, the program is transformed into combinator expressions (containing no variables). Second, these expressions, applied to arguments, are reduced as dictated by the definitions of the combinators.

4.3 Requirements for Supporting Computational Models

To summarize our examination of the different computational models for both functional and logic languages, we outline here a list of capabilities that must be provided when considering implementation, optical or electronic. We find that the basic functions required for the execution of programs in PARLOG and functional languages are quite similar [25]. The key capabilities that must be provided for an efficient implementation of the computational models are as follows:

- Digital representation of data and data structures: Precise comparison of data is required in these computational models.

- Random access or location-based addressable memory: for implementing data structures such as lists and graphs that are modified dynamically. Also, there must be provision for accessing different components of the data structure.

- Dynamic memory management: traditional symbolic computing requires dynamic allocation and deallocation of memory locations since data structures are dynamically created and disposed of.

- Primitive logical and arithmetic operations: some logical, such as Boolean operations, and primitive arithmetic operations, such as addition and negation, must be supported. In symbolic computing logical operations are more prevalent.

- Comparison operation: comparison operation is required for symbolic matching purposes. This matching should be digital due to the level of precision required.

- Multiple tasks will be running, and issues such as multitasking on the same processor need to be carefully investigated and either eliminated or accommodated.

- Recursion must be supported.

The fundamental difference between the support for the computational models for logic languages and functional languages is that the data structures in logic languages may be partially instantiated during execution, that is, their extent are not fully known at all times. This is not true for functional languages.

It is important to reiterate the conclusions on the need of memory. An important implication of the representation of data by graphs and trees is that the representation of the connections between the data is as or more important than the actual data items. Traditional computer designs handle this problem through the use of pointers. Pointers are typically addresses of locations where other data items are stored. This approach to the representation of complex data structures is attractive because it allows complicated relationships to be efficiently stored without having to be specified at the time the program was developed. It also requires that the machine possess addressable memory. A more important result of such a representation, especially in the case of these computational models, is that the access of different parts of the data structure is simplified. The access to different components in a data structure is important since individual components can be modified. A simple example of such a requirement is in the merging of two linked lists, of arbitrary lengths, in a LISP program.

Another important issue that will have to be addressed in an optical computer architecture based on a parallel computational model is one common to multiprocessor architectures. When considering a parallel processing architecture with a finite number of processors, each processor must have the capability of handling multiple concurrent processes. This is especially critical in symbolic processing applications where data structures do not have fixed extent and the number of executing processes can vary, unlike in signal processing where a problem size does not change once in execution. Handling more than one process is equivalent to handling more than one context and requires context switching. Managing context switching in turn requires some form of memory, typically, stacked memory structures. Memory structures such as stacks are difficult to implement without location addressable memory.

Yet another issue that must be addressed in traditional symbolic processing is recursion. Recursive algorithms are often employed by expert system programs and other symbolic processing applications because of their ability to handle problems of arbitrary size. In a typical recursive algorithm, such as the recursive copying of a tree of data, a function or subroutine is called from within itself to solve part of the problem presented to the original calling routine. For this algorithm to execute correctly, a new set of local arguments must be evaluated for each call to the function. Stacks are typically used for this process, and as with context switching, this means that some form of location addressable memory is required.

5. Primitives in Optical Computing

An examination of the optical computing primitives and the fundamental and practical limitations of optics will provide us with data as to how well various computational models can be supported, and what changes must be accommodated to make them amenable to optical computation.

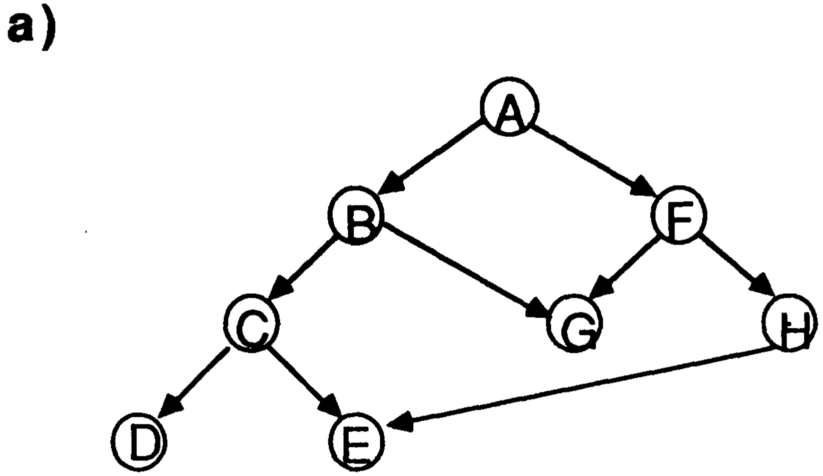
From the optical device point of view, the primitive operations that are required by the target computational models are quite complicated since they must operate on data structures, not on isolated data items. Moreover, because of the dynamic nature of the data structure, the makeup of these data structures is not known until the time of actual execution. This means that the required operations cannot be performed by optical architectures such as simple correlators since the structure, as well as the values, of the data must be examined.

Operations like matching of digital data items could, perhaps, still be performed using correlations. One approach would be to use the correlator to compare simple atomic values. This approach, however, cannot fully exploit the advantages of optical correlation systems. Since the whole item cannot be compared at once, each atomic value would have to be compared separately. This undermines the advantages of parallelism inherent in the optical correlation operation. Secondly, since all bits are individually significant, the correlator would have to be accurate enough to distinguish differences of one bit. This places severe accuracy constraints on the correlator.

Another approach would be to maintain the data structure as a single entity. At present, this type of representation is difficult to achieve in any computer because the data structures change, requiring a means for selecting, adding, deleting, splitting and joining as well as managing the free space required and freed by these operations. We examine why these issues are important and why they might be difficult for an optical computer.

5.1 Data Representation

Although optical computing as applied to signal processing and to neural networks traditionally uses analog data representation, because of noise problems and the requirement of precise representation of data structures in expert systems, digital representation is preferred. A particularly difficult issue in an analog system is how pointers are implemented. Exact representation of data structures can be most easily accomplished with digital representation. Analog representation could be employed if the probability of error were sufficiently low (1 error in 10^{15} operations), but, in practice, digital systems are the only choice. This choice will then limit the types of optical computing



b)

	A	B	C	D	E	F	G	H
A		●				●		
B			●				●	
C				●	●			
D								
E								
F							●	●
G								
H					●			

Figure 5.1 Adjacency matrices

structures to those that represent the data in digital form. This does not, however, imply that all of the computation must necessarily be digital. Very low-level (node to node) matching might be able to use analog methods, but manipulation of the data structures must use digital representation. This conclusion is a practical one rather than an physical one because it may be possible to develop correlation strategies which have the required bit error rate. However, this seems likely only if the domain of comparison is extremely limited. One possible solution to matching is to employ digital representations for the data items but use an analog element to determine if items are identical. Nevertheless, the most important and time-consuming part of the task is the data structure manipulation which does not benefit directly from employing analog elements.

The first item which must be investigated in developing ways to perform the functions for symbolic processing is the representation of the graphs and lists using optical architectures. As stated previously, this type of representation typically requires the use of pointers and location addressable memory. This need for addressable memory can be tackled in two ways: by developing another type of memory structure or by developing a way to implement addressable memory with optical devices. We discuss the issues in implementing addressable optical memory next..

5.2 Memory

Several methods can be employed to represent data structures in digital form without resorting to location addressable memory. Adjacency matrices represent one possible approach. Graph structures can be represented in a matrix structure by assigning nodes of the graph to rows and columns. When there is a connection between nodes, an entry is made at the intersections of rows and columns of the two elements. A directed graph may be represented by using the rows to indicate the node the connection is from and the columns to indicate the node it is connected to. No addressing is required to check interconnections between data items; it is all present in the matrix. To set up the connections, however, some means is required to address and set/reset the elements of the matrix. Figure 5.1 illustrates the representation of a simple directed graph.

While the adjacency matrix appears quite suitable for representing arbitrary graphs, there are certain limitations to this approach. The 8-node graph shown in figure 5.1 has only 9 connections but requires a 64 element adjacency matrix. First, unless the graph is densely connected, the adjacency matrix is quite sparse and therefore inefficient in space to represent the original graph. Second, the adjacency matrix represents only connections, it has no descriptions of the values of the nodes in the graph. Third, and most critical, is the difficulty of handling additions and deletions to the graph in the adjacency matrix. The following example will illustrate the difficulty.

A simple deletion of a node corresponds to removing the row and the column for the graph. Since the matrix is typically of fixed size, after a number of additions and deletions, we would have to squeeze in the graph to exploit the intermediate empty rows and columns or use some other strategy to keep track of the unused rows and columns. In case of symbolic computing, where frequent changes are made to data structures, one would have to have an efficient approach to locate the deleted data locations. In conventional systems this function is often performed by a linked list. However, linked lists require addressable memory so our assumption about not requiring such memory is violated. Because the adjacency matrix has the same problems as location addressable memory and because it is so space inefficient we conclude that it is not a viable option for symbolic computing.

Another attractive method for implementing a different memory and computing structure is the optical finite state machine (OFSM) [13]. The feature of this architecture, unlike in conventional electronic computers, is that the memory is not separated from the processor. Computing systems have been proposed which are composed of parallel planes of 1000 x 1000 optical gates performing the logic operations of the finite state machine [3]. The conventional way to design a finite state machine (for eventual electronic implementation) is to enumerate all the possible inputs, outputs and next states, and then develop some combinatorial logic to perform that function. Such an effort would be tantamount to specifying all of the possible data structures and all the possible values of the data items at the time the machine is designed. It also would require specifying the answers for each possible case. In other words, design of such a system would be identical to enumerating all answers for all of the possible computations the machine could ever make. This is achievable for small systems such as sequencers in microprocessors and other simple controllers, but it is impractical for designing computers which solve traditional symbolic processing problems. In a symbolic processing computer the number of possible states is enormous, perhaps as large as the number of states in the machine (for a 1000 x 1000 array of bits, this is $2^{106} \sim 10^{300000}$ different states).

The other approach to developing a finite state machine is to specify the transition rules for the states in such a way as to avoid specifying all of them explicitly. Symbolic Substitution is such a method [12, 13, 26]. However, it has the disadvantage that the machine is no longer highly interconnected. Only pixels within a certain neighborhood can communicate directly. This will eventually limit the speed at which a computation can occur, since many cycles could be required to transfer data around the plane. The use of shuffle networks [29] improves performance by increasing the connectivity. Symbolic substitution does have the advantage of being easily implemented and may be able to employ high-speed (gigabit) optical components. However, symbolic substitution is not a method to implement memory but a way of implementing "circuits."

One uniquely optical solution would be an all-optical storage in volume holographic media. Such systems have the potential to store large amounts of data in small volumes, but have several problems which make them difficult to use for addressable memory

Page addressable holographic storage is one possibility. However, some means are necessary to generate the readout wavefront. This would not be a trivial task since many independent wavefronts would be required to address many different pages. At present, there are no methods for generating these read beams with the speed required to use them as working storage. Page addressable holographic memories also have the problems with limited page size [16], and with the need to select the appropriate data in the output page.

Associative architectures solve the problem of generating the reading wavefronts. However, optical power considerations still limit the size of the page unless the input pattern is nearly the same size as the output page [16]. As we have shown, it is unlikely that large input patterns would be used to represent data. Thus, a large input pattern would either have to be generated as part of the addressing process, or the associative memory would be used to store small patterns. The use of large pages has the same problems as page addressable storage with regard to selecting the appropriate data in the output page. The use of small pages is problematic since associative architectures cannot store data which have very similar input patterns. This means that with small input patterns, fewer sufficiently different patterns would be available and fewer data items could be stored. This is the same as stating that associative memories are inefficient (in terms of capacity) as compared to normal RAM.

The added functionality of the associative memory does not make up for this reduced storage. This is because of the need for pointers. The representation of pointers is arbitrary and therefore unaffected by the use of a memory which allows input patterns that are different from binary addresses. In broad terms, this means that associative memories cannot be used because they are ill-suited to represent the data structures used in traditional symbolic processing.

The other solution to location addressable memory is to actually construct memory that has binary addresses. The problem with this approach has been the difficulty in generating the decoding addresses. Two approaches have been developed to decode binary addresses [29,31]. Both employ $\log_2 n$ stages, for a memory of size n, to decode the address and work much like the address decoders in common electronic memory chips.

The basic approach in both cases is to construct a decoder which converts n binary bits into a 1 of 2^n bits. Then as in an

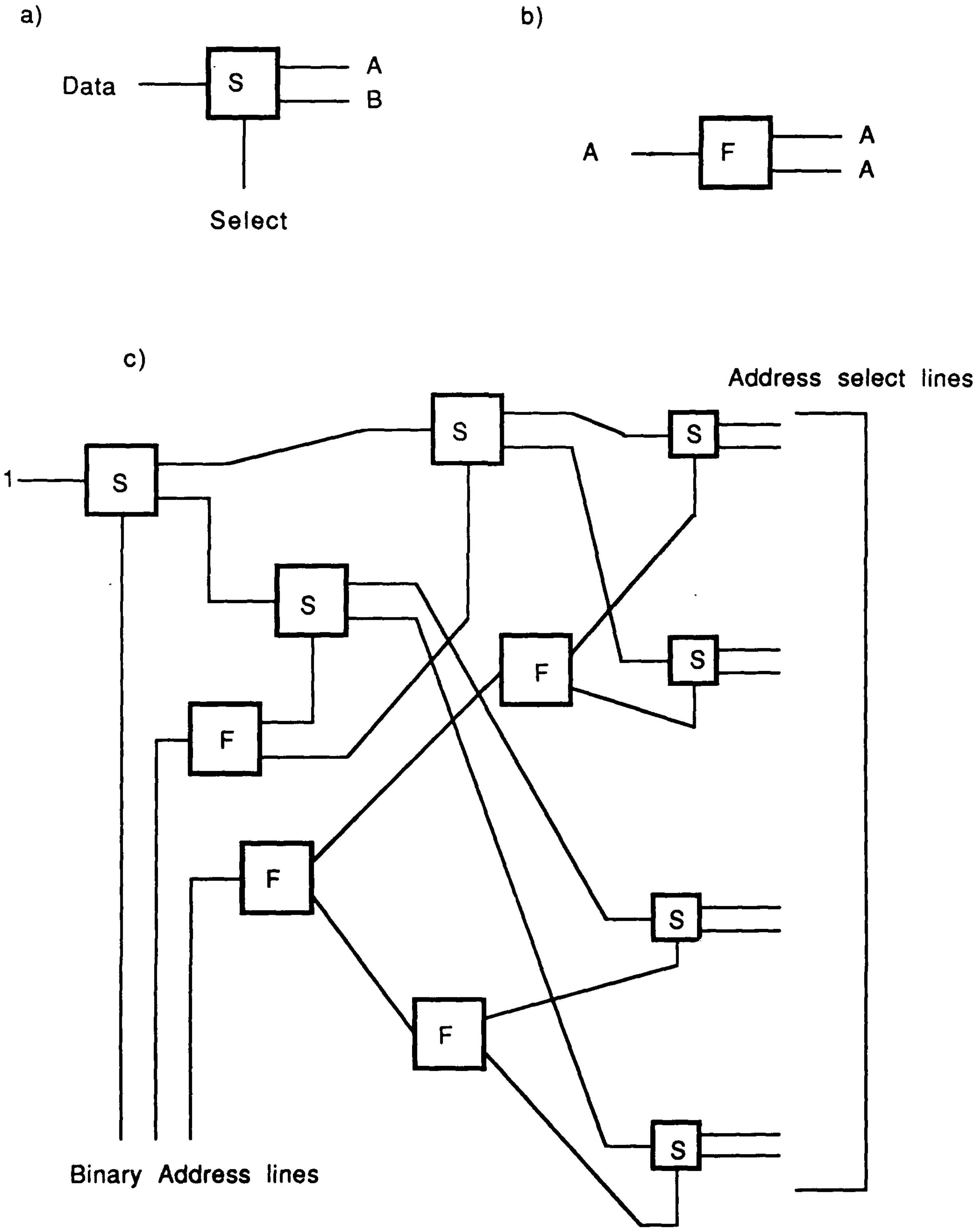


Figure 5.2 Memory decoders

electronic memory, these lines address either the x or y lines of an 2-D array of memory elements. Memory elements are located (but not shown) at the intersection of the x and y address decode lines. When one X and one Y line are activated, the memory element at the vertex is addressed and can be read or written.

Figure 5.2 illustrates how a decoder can be constructed. Two gates, a select gate and a fanout gate, can be connected so that when a binary address is used to set the select lines, a 1 passes to one and only one of the outputs. The select gate is analogous to a single pole double throw electrical switch. The input can be directed to either of the outputs, depending on the state of the select line. The fanout gate is nothing more than a gate with has enough gain to allow it to power two gates.

An example of how a select gate could be implemented is shown in Figure 5.3. Optical logic gates with reflection and transmission characteristics shown in Figure 5.3b are used. When optical power is present only in the bias beam, most of the light is reflected from OLG1 and goes to OLG2. If data is present, it switches OLG2 and causes the beam to be sent to output A. If the select beam is present, then OLG1 is mostly transmitting and the power goes to OLG3. Data could then switch it to send power to the output B. The system described here is designed to be illustrative of the operation of such a memory, rather than be a serious attempt at developing one. In a real system, arrays of devices would need to be employed. The system performance could then be estimated from the time of flight, if the optical logic gates were very fast or from the gate speeds themselves.

The fastest speed for an optical memory would be determined by the time of flight of the system. This is because, while such devices could operate in pipelined mode, the important time for a memory device is the time it takes to fully complete a read or write or the cycle time. For our approach we estimate that the time required to decode the address in a useful memory (>64K bits) would be between 10 and 100 ns. We believe this is too large to be useful in an application which is so dependent upon accessing memory. This speed is not slow by today's standards, but electronic memories are expected to be implemented with speeds of 1 ns by the end of the century. This speed advantage would not be of much import if the optical memory had greater capabilities than the electronic one. However, in this situation the optical system is emulating an architecture which is very well matched to electronic implementation. The optical memory is much more limited in its capabilities.

A final approach would be to employ electronic memory. This has the advantage of employing an efficient, mature technique for storing data. However, it places the interface between the electronic and the optical system in the middle of the most time-critical function, that of manipulating the data structures. The implication of using such a memory is that while a parallel architecture can be designed using basic processor-memory building

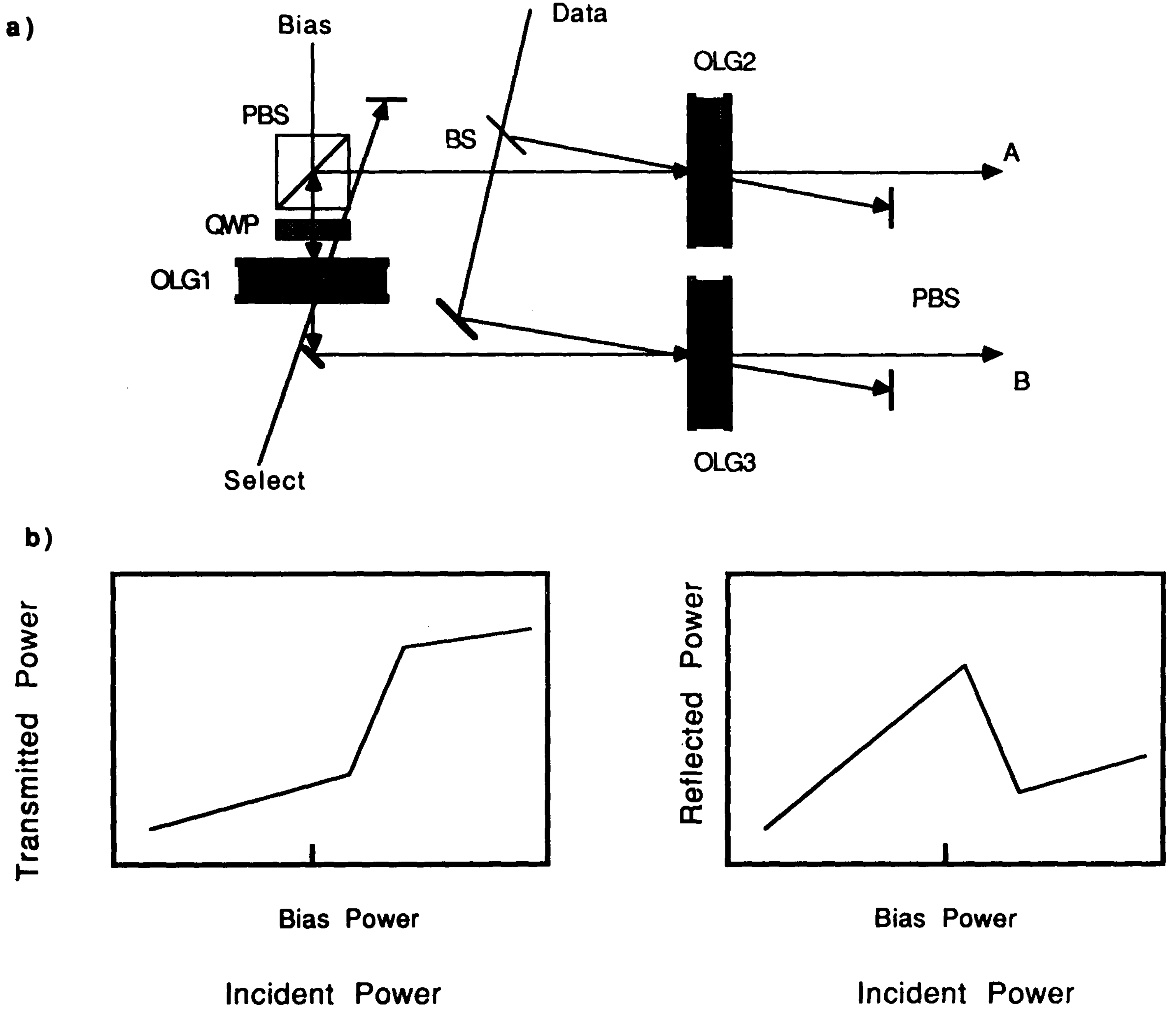


Figure 5.3 Optical select gates

blocks, each building block is a von Neumann architecture with an optical CPU and an electronic memory. This means that, to be a viable option, the optical processor must have sufficient computational power to overcome the disadvantage of the interface and to also fit well with a traditional electronic location addressable memory. These requirements are not likely to be met when compared to an electronic CPU. The simpler RISC architectures already have a bottleneck with the memory access rather than with the processing. This state of affairs is likely to continue.

Another approach to employ electronic memory would be to develop an architecture which has a limited amount of all-optical memory as a cache memory but uses electronics to store the majority of the data. However, this means that the need for an all-optical addressable memory has not been eliminated, just reduced. It also means that a complex system is required to manage the cache memory. This may or may not be a viable approach for optical symbolic processing, but the need to address very different parts of the memory, and the need to develop a system which can do so in parallel does not make this option very attractive.

5.3 Interconnections

The complexity of the interconnections between parts of an architecture significantly affect how efficiently a problem can be mapped onto it. If an architecture has dynamic interconnectivity, then it becomes possible to represent the connections between data items by interconnects in the processor. In this section we examine the physical limitations to optical interconnects which preclude the use of such dynamic interconnects for symbolic processing, and which put limits on the complexity of the static interconnects which can be employed.

The interconnection schemes which we would like to consider are shown in Figure 5.4. We would like to develop a system which could interconnect any element of the input plane and connect it to any element in the output plane. This system has been extensively investigated [11-17, 32, 34] and the two main approaches are to employ a thin array of deflecting elements just after the input plane or to employ a volume holographic element. Both systems offer advantages and disadvantages.

The specific problem which we examine is the connection of large input and output planes (10^6 elements) in a way which is as general as possible and would allow multiple simultaneous updates of the interconnections at speeds which are comparable to the memory access speeds. In this system high speed memory access would not be as important since the connections would be already in place and fewer would be required. The implementations of such systems with thin and volume interconnects are slightly different. We examine each independently starting with thin holograms.

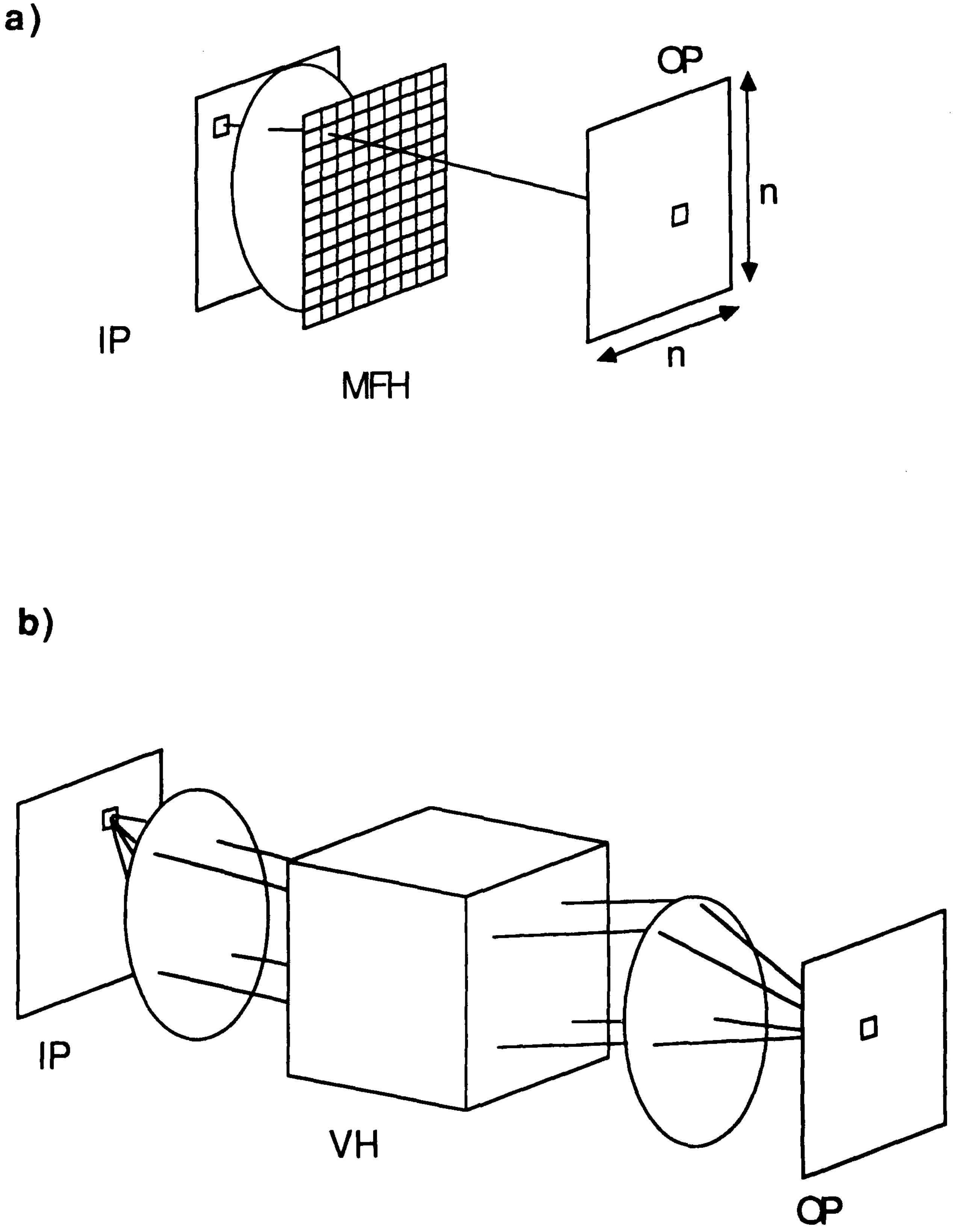


Figure 5.4 Interconnection schemes

It is clear that a multifaceted hologram [11] would be able to perform the operations required for the thin interconnect if a material could be found which responded fast enough. The number of interconnects which can be performed by the interconnect is proportional to the space bandwidth product (SBP) of the hologram. Another issue for this system is the total system volume or the total volume which the interconnect system from the input to the output plane would require. To determine this volume, we examine the diffraction of light emanating from one element of the input plane.

We assume that collimated light emanates from the input plane. This is equivalent to assuming that a plane wave is incident on a transparency which has elements transparent or opaque depending on the data to be transmitted. We also assume that the first element in the interconnect is a thin lens which without any intervening elements would focus the light from all input elements to the center of the output plane. By separating the functions of focussing and deflection, the problem is greatly simplified. Since the only function of the hologram is deflection, it can be modeled as a simple grating. For a phase grating, it is well known that a large diffraction efficiency can be obtained. This means that in the output plane, the output will essentially be the spot pattern from the aperture of the input plane, or equivalently in this analysis the multifaceted hologram.

Assuming square elements, the problem can be examined in one plane. The amplitude of the light incident on the output plane is just:

$$u(x) = A\ \mathrm{sinc}(a(x-m\ s_1)/lf) \qquad (1)$$

where s_1 is the center to center separation between input elements, a the size of an input element, m is the element number, f is the distance between the planes, and A is a constant related to the light intensity. The distance d between the first zeros of the spot in the output plane is

$$d = 2\ f\ l/a \qquad (2)$$

The distance f is limited by the maximum deflection obtainable from the thin lens-deflector combination. For a holographic situation, this is approximately 45° [11]. If s_2 is the center to center spacings between the elements on the output plane, then the minimum separation is given by

$$f = n(s_1 + s_2)/2 \qquad (3)$$

Solving (2) and (3) simultaneously we find that

$$k_1/d + k_2/a = 1/nl \quad (4)$$

where $s_1 = k_1a$ and $s_2 = k_2d$.

For close packing ($k_1 = k_2 = 1$) this equation gives the relationship for the smallest elements. It is easy to show that the smallest system volume and components results when $a = d = nl$. Since there are n x n elements in the deflector, the dimensions of the input plane, output plane, deflector and distance between them are all n^2l and the required SBP is n^4. This means that with plane sizes of 1000 x 1000 and light in the near infrared, the planes have dimensions of almost 1m x 1m. Notice that in this configuration it is not possible to use the total theoretical SBP of the hologram. Moreover, the volume required per interconnect would be

$$\text{Vol/interconnect}= n^6 \, l^3/n^2 = n^4l^3$$

For a 1000 x 1000 plane, the system volume would be nearly $1m^3$. Clearly, it is not possible to use direct deflection by multifaceted elements to obtain arbitrary interconnects that will have good operating performance (the transit time of such a device is over 3 ns, not including the skew associated with the deflection).

If a programmable interconnect were to be used, then it would have to be totally general and would have to be the same size as an arbitrary fixed interconnect. One possible way to reduce the complexity of the interconnect would be to structure the interconnects [14] so that only a few interconnect patterns would be possible. In this way, it is possible to get some limited degree of programmability at the cost of having only a few degrees of freedom. This, clearly, cannot be used to connect arbitrary data items together, our initial goal, and thin deflectors must be eliminated.

It has been shown [15,17] that volume holograms have a much smaller volume requirement and that the input planes can be much smaller. However, it is difficult to use these holograms for the dynamic interconnects required. Current photorefractive materials have problems meeting the speed requirements if good diffractive efficiencies are to be maintained [33]. The creation of intermodulation gratings will limit the ability of these systems to have the interconnects updated in parallel [16].

5.4 Processing

Several approaches have been previously described for processing symbolic data [4-10], but most have used computational models which are very different from those used in traditional symbolic processing. These approaches to optical architectures are not applicable here since they put strict bounds on the types of problems which can be solved.

Optical finite state machines would appear to offer the greatest computational flexibility, but as we have already noted are impractical for large problems. Several other approaches which employ the basic structure of the FSM are optical sequential logic [34], symbolic substitution [27], and the optical clocked architecture [35]. All of these have the capability to perform arbitrary operations and symbolic substitution has been shown to be a general computer in the sense of a Turing Machine.

We have shown that the most basic required operations for traditional symbolic processing applications are data movement and comparison, with the understanding that these will be employed to manipulate and compare data structures.

In summary, it is possible for optics to perform most of the memory functions needed for symbolic computing, but it is unclear that optics has any clear advantage over electronics except for providing a large degree of connectivity between the computing elements and thus providing a source for increased throughput. From this analysis we believe that only a FSM-based processing scheme such as symbolic substitution has the computational power required to implement the tasks required for symbolic computing. However, since memory is difficult to implement this way, in the next section we examine the issues involved in implementing logic languages and functional languages using optical primitives by examining example optical computer architectures with the goal of minimizing the amount of memory required.

6. Optical Implementations of Symbolic Processing Languages

We consider the issues in implementing the two categories of languages, functional and logic, in optics.

6.1 Issues in Optical Implementation of PARLOG

Based on the AND/OR graph reduction computational model and the abstract parallel architecture [23], which appears most appropriate for PARLOG, a broad optical architecture for PARLOG was attempted.

As described earlier, each PE in the parallel abstract machine must contain a number of computing functions or computing agents, such as handling variable bindings through matching, managing the AND/OR process tree, and handling message communications with other PEs. The interconnection of the PEs could be handled very simply though point-to-point optical connections. However, implementing the PEs is much more complicated. Optically, the computing agents could be realized in terms of a cluster of OFSMs which execute the algorithms comprising the function of the agent. The PEs, as well as the agents, communicate via messages. Different message types can be recognized by the use of an associative pattern matching on the message type. Since OFSMs are limited in complexity, all control operations, such as logic, matching and arithmetic operations, are done external to the basic finite state machine as in a traditional CPU. Another approach would be to use symbolic substitution to perform all of the logic, arithmetic and matching functions.

The crucial design of the optical architecture, however, is the data representation of the DAGs, as we saw earlier in the context of optical memory. Given the severe limitations of optical memory, linked list structures cannot be implemented as easily as in electronics. The simplest solution is to represent the DAG as an adjacency matrix despite space inefficiencies. The adjacency matrix will only contain topology information of the DAG and will need augmentation to represent information about the node values. More important than the necessity to represent values of the nodes in the DAG is the fact that a node in a DAG may be partially instantiated, that is, it possesses only a partial structure, and waits to be fully instantiated, possibly through input matching with some other variable not yet evaluated. Such a situation may require waiting in a demand list of another variable to avoid busy waiting on a variable by a PE. Moreover, when the node does become completely instantiated, it may assume some structure already present in the memory. This would then require that the matrix memory be modified to point to the data structure and the connections to the demand list be removed. This removal would then require either the rearrangement of the total memory or the addition of the free or deallocated space to some list. In any case, efficient representation of partially instantiated variables, common to logic programming languages, implies extensive use of pointer structures and location addressable memory.

Because logic languages use coarse-grained control-level parallelism, and because of the difficulty in feasibly implementing data structures and required computing agents in the PEs of the architectures using OFSMs and separate memory structures, we eschewed the possibility of implementing optical processors for executing logic programming languages. Comparisons reveal that functional languages provided a better alternative for optical implementation.

6.2 Issues in Optical Implementation of Functional Programming Languages

One of the major disadvantages of logic programming language execution in optics was the presence of partially instantiated variables since this caused an extra burden in data representation. Functional languages, on the other hand, do not present this problem since in their computational model, all variables are either not at all or fully instantiated. To evaluate functional languages for optical implementations, both reduction and dataflow models were evaluated since both have been implemented in electronics for high performance. The data flow computational model is considered first followed by a discussion on graph reduction.

A dataflow machine [22] requires multiple processors to operate on different portions of the dataflow graph if and when input arguments for different operators become available. This requires each PE to maintain the complete graph and possess the capability of recognizing input arguments and their context with the help of tags. Since an operation may require multiple arguments, a PE in the multiprocessor architecture has to maintain a tag matching unit while waiting for all arguments of an operation. A typical topology for such an architecture is an n x n routing network connecting the PEs [22].

From the point of view of optical implementation, the dataflow computational model leads to a coarse-grained parallel architecture since each PE has to be cognizant of the complete dataflow graph (as in the case of PARLOG and its local memory for the compiled program). However, the overhead at the level of data object management is also substantial since tags must be generated, maintained and matched at runtime. In a broad sense, the complexity of the optical architecture for this computational mode will be as complex as that in a logic language except that the variables will not be partially instantiated during evaluation.

The next computational model considered is that of reduction. While the complexity of graph reduction is typically similar to that of data flow, one flavor of graph reduction that appears most promising for parallelism is combinator graph reduction (CGR) [20]. In CGR, each step is an atomic step in which the graph is mutated in a manner consistent with the reduction rule of the corresponding combinator. In a distributed system where the graph is distributed in a network of PEs, a message-passing strategy will allow each reduction to occur in piecemeal fashion. The graph reduction evaluation model therefore appears well suited for a fine to medium-grained parallel architecture in which reducible expressions (redexes) are evaluated in parallel.

The critical design challenge in optics is how to realize the combinator reduction process in parallel. Since the PE responsible for the combinator application is a simple combinational function, an OFSM implementation is not necessary; however, since the argument of the combinators can be a data structure (a list in the general case) of any size, the transformations may be difficult to handle if the data is moved every time into different nodes. In the electronic case, pointers can be used very conveniently without actual movement of data. Thus, unless the data is of simple structure, using pointers becomes attractive.

Another instance where pointer structures are necessary is in the evaluation of common subexpressions. To avoid wasted computation, common expressions are shared in graph reduction unlike in string reduction, which is similar in all other respects to graph reduction. However, the use of shared expressions in CGR implies using indirection to ensure that argument values are not lost before all expressions involving the subexpression have been evaluated. Given these issues, it would therefore appear more attractive to examine nondistributed architectures, where graph mutations are managed in a common large memory. Such architectures would be designed to exploit low-level parallelism in optics. Our design of a symbolic processing optical architecture, SPARO (Symbolic Processing Architecture in Optics), is based on these motivations. We provide a brief overview next. We have presented the results of our analysis in designing optical expert systems and traditional symbolic processing architectures in this paper. We have attempted to examine the complete problem of symbolic computing by systematically analyzing the computational models required for symbolic processing rather than attempting to construct smaller subsystems of symbolic processing in optics. We have reached some critical conclusions regarding the use of optics to solve these problems. In particular, we have accomplished the following goals:

We have employed a top-down and bottom-up approach to design an optical symbolic architecture. In the top-down analysis, we have abstracted the requirements necessary to support the implementation of symbolic processing languages and their computational models. Special emphasis was placed on parallel computational models such as graph reduction and data flow. We have then derived a list of requirements that must be met if these computational models are to be implemented. In the associated bottom-up analysis of optical computing methods and techniques, we have looked at their strengths and weaknesses to determine how optics can be used to meet the top-down requirements.

Based on our analysis, we find that no single optical computing scheme matches the requirements for symbolic processing. This is because of the crucial need for a good representation and manipulation of complex, dynamic, structured data. Such a requirement is best addressed by the use of location-based or addressable memory, a feature that is not well-met by current optical technology. We conclude, however, that computational models for fine-grained parallel execution of pure functional languages appear more amenable to optical processing because the need to communicate and maintain several different types of data structures is less severe when compared to other computational models and other programming languages.

In summary, we have shown that symbolic processing is a complicated and difficult task which may or may not have an optical processing solution. Only a critical performance evaluation of the final architecture will the determine the validity of this approach to optical processing.

7. SPARO: Symbolic Processing Architecture in Optics

In the previous sections we have attempted to describe why optics can or cannot do operations required for symbolic processing. The purpose of this was, of course, to find what could be done to construct an optical symbolic processor. In this section we draw the conclusions of the previous sections together to outline how such a processor could be constructed.

SPARO was developed to meet several criteria. These include scalability of the architecture, exploiting the parallelism of optics, and the ability to perform tasks required to solve real problems. In the following short description of the proposed architecture, we only provide an overview of the processors that constitute SPARO and how CGR is executed in parallel.

7.1 Processor Design Outline

One of the major obstacles overcome by the SPARO architecture is the difficulty of implementing addressable memory with optical components. This is important since the design of the processor for CGR is intimately tied to the design of local memory in a nondistributed architecture.

The conclusions that location-based addressable memory is difficult with optics technology and that pointers are essential are at the heart of the architecture of SPARO. These constraints lead to an architecture that employs fine-grained processing elements or nodes (as called henceforth because of their analogy to nodes of the graph and to nodes of a network) in a nondistributed architecture. The processing elements are very tightly coupled, and relatively simple communication, i.e., a few data words, are used. Also, the memory is not separated from the processing elements and thus can be considered to be a part of the nodes. This structure meets the requirements imposed by optical implementation by reducing the need for addressable memory while still offering pointer-like constructs. It does this while exposing the parallelism present in CGR.

The combinator graph (CG) has a very special structure. Each node in the graph has an 'arity' or out-degree of two. The only difference between the CG and binary tree is that nodes in the CG may be shared by more than one parent node (all nodes can only have two child nodes). This limited arity of nodes in the CG suggests a direct mapping of the graph onto the processing elements. The advantage of this mapping is that each node as memory only needs a small amount of storage since it only has to maintain information about itself, its two children (more than two child nodes can be easily accommodated in this scheme), and its current parents. As a processing element it also contains information regarding its state and some very limited working storage.

Unlike other non-Von Neumann architectures, SPARO has no addressable memory (in the traditional sense) anywhere. The only memory is the set of few registers in each node. These registers serve two critical functions in SPARO. First, they are used to build data structures. This is because the registers are designated as pointers to other nodes rather than for containing their actual values. Data structures are constructed by linking together processor nodes through the use of pointers. Second, the registers are used to implement message-passing between processing elements. The values of the pointers, the register contents, provide the addresses of messages that are sent between nodes of the CG. Addressable memory is thus emulated by sending messages to where data resides, rather than by fetching memory location.

To perform graph reduction in a distributed manner, the nodes perform primitive operations necessary for executing a reduction and communicate by sending messages. Processing in a node is activated when it receives a message while it despatches messages to other nodes to initiate or complete graph reduction operations. By passing select messages about the types of operations, and by designing the processing elements or nodes to specifically perform graph reduction, short messages can be used to control the flow of the reduction.

A separate interconnection network is used to pass these messages (with data) from node to node. The nature of the architecture requires that the external interconnection network transfer data to and from nodes in the array. Locations of interconnected nodes cannot be guaranteed to exhibit locality in the array, due to the nature of graph mutations. We have therefore partitioned the processor and the communication into two separate functional parts to solve the connectivity and data movement problem. The interconnection network could be implemented in any fashion in so far as it allows the transfer of multiple parallel messages.

If optical elements having 1000 x 1000 pixel elements are employed, then an efficient representation is a linear array of nodes. Each row of the optical element would store the information for one node, as shown in Figure 10. This fits well with the use of a limited number of registers as storage. The use of a linear representation is important since both the connections between nodes as well as the contents of the nodes can be easily represented. The use of a linear array is also attractive since the complexity of the optical interconnects will be lower than for a structure that employs a 2-D array of nodes.

Each row of the array would correspond to one processing node. Each node would consist of several registers to store the information about the combinator graph, a set of bits which would be used to denote the state of the node, and registers which would make up an interconnection network to the rest of the processing nodes.

Examination of optical computing structures has revealed that the most promising concept for realizing the necessary functions needed for SPARO is symbolic substitution or another FSM type architecture. However, because of fanout considerations, it is difficult to implement register to register transfers. This limitation is present because it is difficult for one bit to control the state of many other bits without requiring many stages. To circumvent this weakness, SPARO uses another simple technique to move data between registers, while still employing FSM-like structures for the control and logic operations.

The strategy for moving data between registers and nodes which we consider is a technique called gated interconnects. The approach uses limited, regular interconnects that have optical logic gates at the inputs and that are controlled by pixels of the control logic system. Thus, by enabling or disabling the logic gates, information can pass on the interconnect from one point on the plane to another point on the next plane. These interconnects would likely need to be implemented differently to solve problems with fanout of optical devices [17]. This sort of interconnect is really a special case of interconnections required by all approaches to FSM type computers. Since it is so important, we consider it to be an additional architectural feature to emphasis its importance and to delineate that it may use a different implementation technology than the logic functions. This is possible since at this stage of development gross performance analysis can be done without exactly specifying the total system configuration.

Physically, the operations of SPARO would occur in three dimensions. Planes of optical logic gates and of interconnects allow information to be transferred through the planes of the system. For simplicity, the architecture is shown as if there were one main processor plane which makes use of the functions of the other planes. The other planes are considered to have interconnects that connect the outputs of the registers and gates in the processor plane and carry out the intermediate steps required for symbolic substitution. In a more realistic implementation of SPARO, the functionality of the processor and interconnect planes would likely be distributed throughout some volume so as to reduce the total system size.

7.2 Analysis of the SPARO Optical Symbolic Computing Architecture

We now present a more comprehensive analysis of SPARO. Only certain computational models are feasible for optical implementation. Thus, functional languages executed using fine-grained graph reduction, or combinator graph reduction (CGR), were found to be the most suitable. We also discovered a basic disadvantage in using available optical computing techniques to implement any computational model. This was due to the lack of an optical addressable memory that could compete with electronics. Unlike numeric processing, addressable memory is crucially important in symbolic processing, which requires frequent manipulation of dynamic and structured data such as lists and trees. The problem of handling large data structures as a whole was alleviated if the data as well as the program was represented and executed in a fine-grained distributed fashion as in CGR.

We explore the results of pursuing that conclusion, present the resulting architecture, SPARO (for Symbolic Processing Architecture in Optics), and discuss the lessons we have learned. While SPARO has been designed to exploit the strength of optics, its detailed implementation has not been pursued. However, as our results and conclusions will show, the level of detail in the design is sufficient to perform a gross evaluation of the architecture and to make significant observations as to the advantages and disadvantages of using optics. Most importantly, the design and critical evaluation of SPARO provides us guidelines and recommendations as to where optics can best be used in parallel computing architectures.

We outline the computational theory behind CGR. We then introduce the control flow or execution model of SPARO. Next we describe the details of the computer architecture, that is, the organization and operation of the processor array, the network and the network processor in SPARO. The optical implementation of the different components of SPARO is suggested, and the results of the performance evaluation of SPARO are given.

7.2.1 COMPUTATIONAL THEORY BACKGROUND

From previous analysis [36], we had concluded that CGR should be implemented in an optical computer since the distributed representation alleviated the problem of manipulating large complex data structures. In this section, we briefly describe combinator reduction so that its use in SPARO is more comprehensible. A more detailed explanation can be found in [20] and [37]. CGR is a version of the graph reduction computational model first proposed by Turner [20] for reducing purely functional languages (FP, SASL, pure LISP subset). The combinator graph (CG) or, equivalently, the combinator expression of a given program is really the object code of a program, typically written in a functional language. To use the CGR computational model, the high-level language program must be compiled into the CG.

Implementation and operation of such a compiler is complicated and presents many challenges such as the allocation of processors to the combinator nodes and the identification of recursive code. We have not considered the design of the compiler. Instead, we have focused on the design of the runtime system for CGR, since it is expected that applications written for SPARO would require frequent execution but infrequent compilation. In CGR, special operators or combinators are introduced to allow the bound variables of a function to be abstracted [20]. This leads to efficient representation and execution of the program, especially in the case of functional programming. The compiler, having removed all the variables, passes a binary graph structure, the CG, to the runtime system, which subsequently reduces the program by applying the necessary variables as arguments. During reduction, the combinators embedded in the CG specify the order in reducing the graph. Three basic combinators are required, although often more are defined to simplify the CG. These are S, the distribution combinator, K, the kill combinator, and I, the identity combinator. Optional combinators such as B and C, which can be expressed in terms of S, K and I, are introduced to reduce and optimize the size of the compiled graph. In conventional notation, the combinators are defined as follows:

$$\begin{aligned}
&S\ f\ g\ x = (f\ x)\ (g\ x)\\
&K\ f\ x = f\\
&I\ x = x\\
&B\ f\ g\ x = f\ (g\ x)\\
&C\ f\ g\ x = (f\ g)\ x
\end{aligned}$$

where f and g are some functions, either composite or primitive, and x is a subgraph to be evaluated. In this notation the functions operate on operands to the right, that is, they are left associative. Figure 7.1 graphically depicts the application of the S combinator. Notice that the apply nodes (shown as empty nodes) provide the required "glue" to hold the graph together. Figure 2 shows an example combinator expression for the function, $f(X,Y) = X^2+Y^2$ The corresponding combinator expression E, shown in the CG form, is:

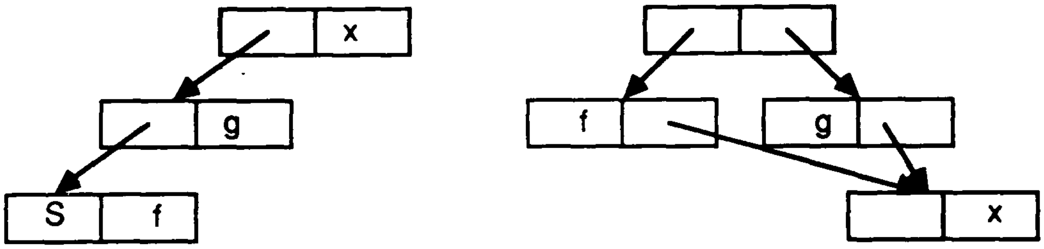
$$E = C\ (B\ S\ (B\ K\ (B\ +\ (S\ *\ I))))\ (S\ *\ I)$$

The reduction is done by applying first X and then Y, that is, by evaluating the expression Exy. Some steps in the reduction are shown in Figure 7.1. Note that all variables, X and Y, are applied at the top of the graph and do not have to be embedded within the graph. This graph structure is possible due to the variable abstraction possible when deriving the combinator expression.

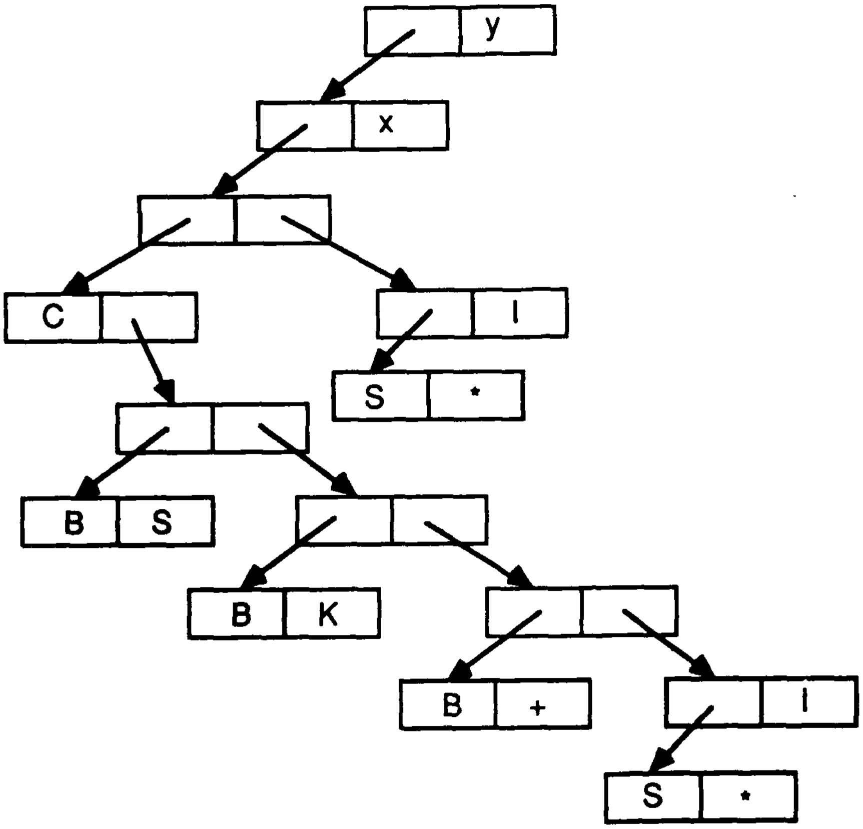
7.2.2 Combinator Graph Reduction

The fundamental difference in the execution of CGR compared to other compiled code is that there is no distinction made between the program and data. The complete program is a combinator expression that is progressively reduced by applying the reduction

S f g x -> f x (g x)



Combinator graph and reduction for S



Combinator graph for $x^2 + y^2$

Figure 7.1 Combinator graphs

rules. Two kinds of reductions take place, that of the combinators, defined earlier, and those specified by the primitive functions such as PLUS or TIMES.

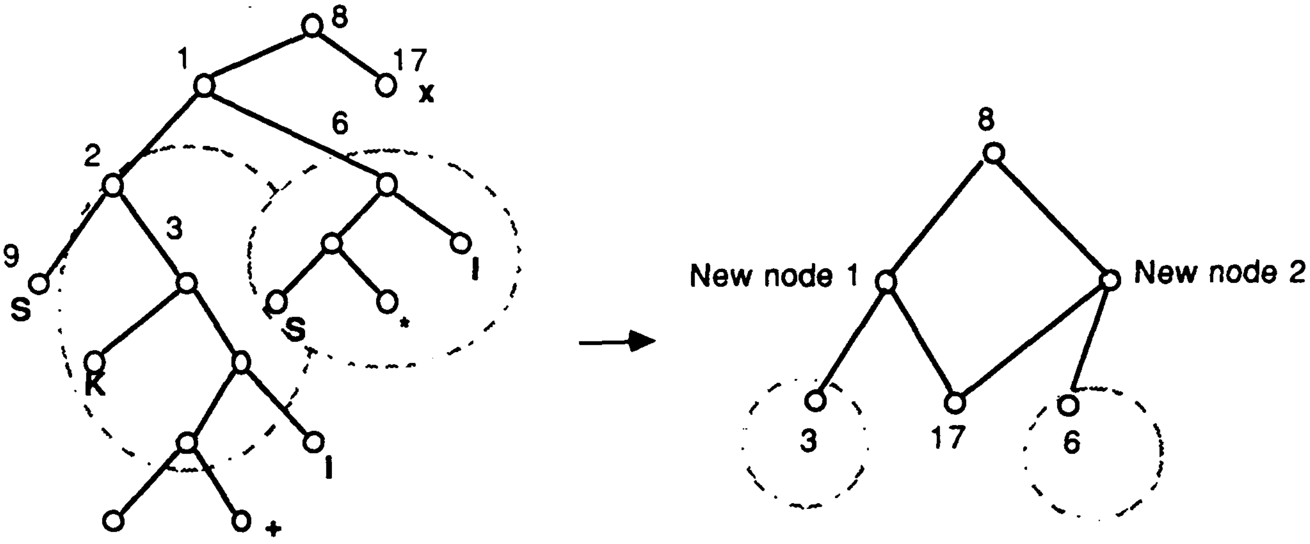
One of the most interesting properties of CGR (and graph reduction in general) is that many reducible expressions (whose arguments are available), or redexes, can be reduced in parallel. In fact, by the Church-Rosser property [37], the final result of a combinator expression in the normal form is independent of the order in which the redexes are reduced. However, simultaneous reduction of all redexes is most often computationally wasteful since certain subgraphs of the CG are not required in the final computation. An example of such a computation is the evaluation of both subgraphs of an if-then-else expression where only one is necessary. To avoid computing all redexes, an ordering is therefore imposed on the reduction process. Two common strategies in the reduction order are 'innermost first' and 'outermost first' or 'normal order.' The outermost or innermost first refers to the position in the CG where the reduction is initiated. It can be easily shown that the 'innermost first' strategy is not safe [37]; so 'outermost first' or normal order CGR is always used.

The parallelism in the execution of a CG is limited by the inherent parallelism available in the application as well as by the ordering imposed by normal order evaluation. One way to increase the possible parallelism without severe wasted computations is to employ eager evaluation in the case of 'strict' functions. An expression is defined to be strict in a given subexpression if reducing the whole expression requires reducing that subexpression. Examples of basic functions that are strict in their arguments are PLUS and TIMES. However, the amount of parallelism available from the strictness of primitive functions is modest. The amount of parallelism in a program can be improved through a compile-time analysis that identifies all strict functions and the arguments in which they are strict. To exploit the parallelism, each processor node in SPARO is designed as a finite state machine that can execute a specific set of instructions in normal order as well as distinguish between strict and non-strict functions.

7.2.3 Execution Model of SPARO

To describe the SPARO optical architecture, we first present the method by which parallel execution or reduction is induced on the CG, which is distributed among a large number of processors. This method, called "instruction passing", shifts the focus of control to the processor that contains the relevant data. In many ways it is like a data flow architecture, except that short messages, rather than the data, initiate computation.

First reduction step in f = S K ((S +) I) (S * I) representing f(x) = x2 + x**2**



Message/Instruction Passing sequence for S (node 9) reduction:

Message no.	Source	Destination	Instr.	Data	Comment
1	8	1	Apply		Send evaluation request down to leftmost leaf node
2	1	2	Apply		
3	2	9	Apply		
4	9	2	Eval-complete	S	Start S reduction
5	2	1	S1	3	Initiate 1st step: S1
6	1	8	S2	3, 6	Initiate 2nd step:S2
7a	8	free	Allocate	3, 17	Request allocation of a free node with data as children
7b	8	free	Allocate	6, 17	
8a	New node 1	8	Alloc_reply	3	
8b	New node 2	8	Alloc_reply	6	
9	8	New node 1	Apply		

Figure 7.2 Instruction passing sequence for X^2+X^2

7.2.4 Instruction Passing

In instruction passing, an executing macro-function initiates the next macro-function by sending a message to the node with the appropriate data. Figure 7.2 shows an example of an actual instruction-passing sequence for the first step in reducing a simple algebraic expression, $x^2 + x^2$. Nine messages are required to implement the S reduction through instruction passing. The nodes of the graph constitute simple processors c: an abstract multi-processor system connected in a tree as shown in the figure. While this example illustrates the execution model of CGR employed in SPARO, actual implementation issues are considered in detail in the next section. Here we explain how the reduction proceeds using instruction passing in normal order.

To reduce the expression, the runtime system initiates the evaluation by passing an APPLY instruction to the root node 8. Since node 8 is not a leaf node, it sends an APPLY instruction to its left child (LC) node 1. This APPLY instruction sequence is generated until node 9 (S) receives the APPLY instruction. Since node 9 is a leaf node, the processor element of node 9 will then initiate the S reduction.

Two actions take place in node 9. A message is sent to the parent node 2 to collect the arguments of the S combinator reduction. This is expressed as a Eval-complete instruction for an S combinator. The information that the combinator (or function, in the general case) is an S is encoded in the data field of the message. While the message is sent to the node 2, node 9 is deallocated or freed. Node 2 continues the S reduction process by sending the instruction S1 to its parent node 1 with the its right child (RC) address, node 3, as one of the arguments for the S reduction. Node 1 then continues by sending instruction S2 to node 8 with both arguments of the S reduction, namely, nodes 3 and 6. Note that 3 and 6 represent the composite nodes that are unevaluated as shown in the figure. To accomplish the reduced graph structure shown on the right, node 8 allocates two new nodes, New node 1 and New node 2, in parallel. At the end of the S reduction step, node 8 initiates the reduction of the reduced graph by sending an APPLY instruction to its LC node, New node 1.

The message passing approach in SPARO promotes the efficient implementation of distributed control since it does not require any shared program or data storage. Each node processor in the evaluation process of the above reduction step acts independently and has its own control. It does not have to store or manage any data besides its own. By designing each node processor as a finite state machine with a minimal specified instruction set, the set of nodes constituting the parallel processor, any general CG can be reduced. All parallelism in the application, exposed by the graph, is obtained by exploiting strictness of the functions evaluated. Thus, when new nodes are to be allocated by node 8, two messages, 7a and 7b, are sent out in parallel. Similarly, any operation, such as PLUS, would request the reduced values of its child nodes in parallel by sending out two messages in parallel. The evaluation of those arguments in turn may require further parallel evaluations. Thus as the graph reduction proceeds, the parallelism in the application unfolds.

Generally, the degree of parallelism in execution increases when the instructions reach deeper into the mutating CG where more reductions can occur simultaneously. However, the price of this parallelism is that many instructions need to be passed between processors. In a system where the processors execute much faster than the communication network that delivers messages, this type of computing would normally produce a serious bottleneck. However, the communication burden is alleviated by the fact that fine-grained processors need very simple messages: addresses of the source and destination, an instruction, and at most two data items.

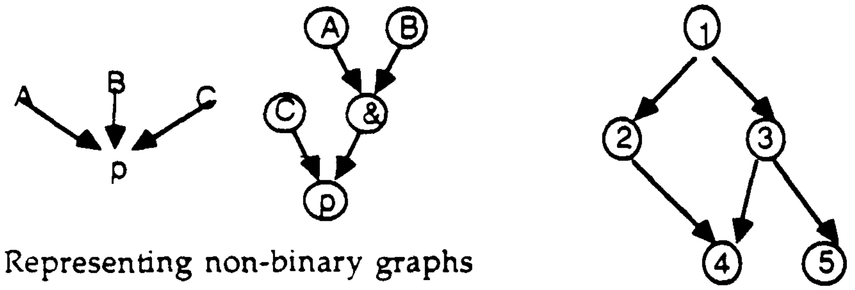
It must be noted that the motivation for implementing instruction passing is essentially to emulate addressable memory on a large number of fine-grained processors. This motivation is provided from our conclusion that addressable memory is critical in implementing well-known computational models.

To implement instruction passing, each processor performs a basic set of macrofunctions, comprising the instructions that are passed. These macrofunctions include instruction sequences for the S, K, I, B, and C combinators, arithmetic, logic and conditional operations, control instructions such as APPLY, ALLOCATE, DEALLOCATE, etc. Using a relatively small set of instructions for each processor node, any relatively complex set of higher order operations can be performed using instruction passing. This is in keeping with the properties of optics since the advantages of fast switching and massive interconnects should lead toward complex computing structures from simple computing elements.

As evident from our design approach, the control and memory do not have separate loci of operation. This is inherent in the nature of the CG construction. An important feature of the SPARO architecture is the mapping of an arbitrary CG onto a fixed array of processors, which is examined next.

7.2.5 Mapping Combinator Graphs onto SPARO

A number of assumptions regarding the representation of functional programs have to be made before the mapping is derived. We have assumed that the typical LISP CONS [38] operation, required in constructing lists, trees and other complex data structures in functional languages, is implemented as a primitive function rather than using the P combinator [20]. Use of the CONS construct greatly simplifies the control of the processor nodes. An ɾxample of how



Representing non-binary graphs

1	2	3
2	-	4
3	4	5
4	-	-
5	-	-

Figure 7.4 Graph representation in two dimensions

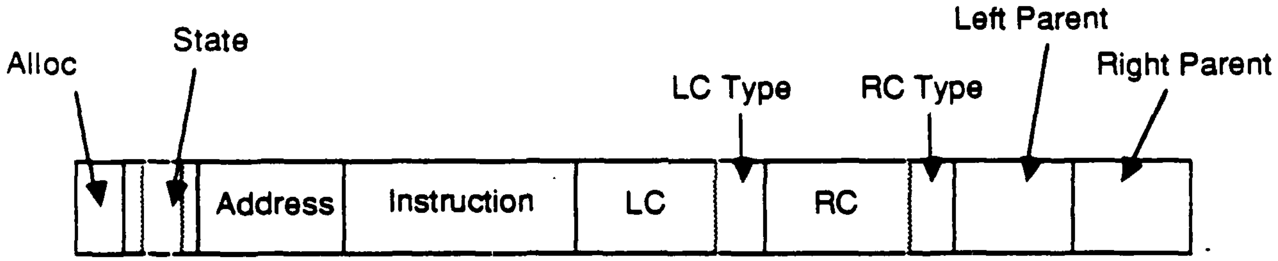


Figure 7.5 SPARO processor node in one dimension

Figure 5

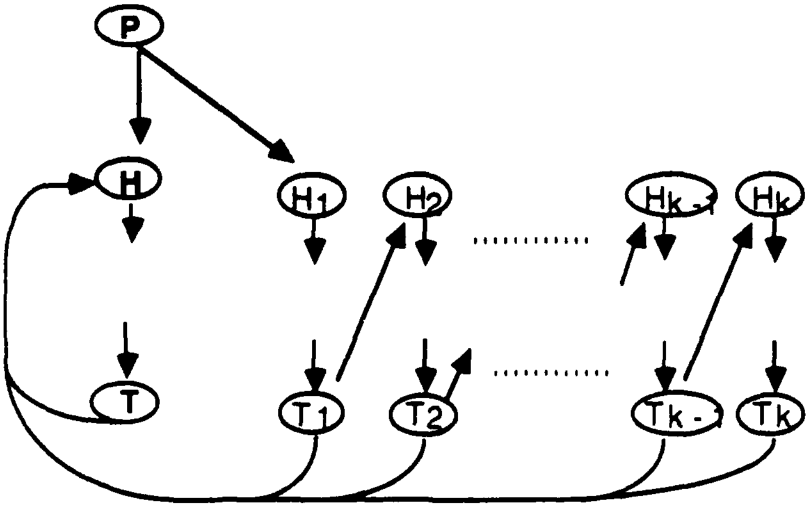


Figure 7.6 Ring network

the CONS operator is used to construct a list of three elements a, b and c is shown below in shorthand LISP notation:

(a b c) will be written as (CONS a (CONS b (CONS c nil)))
where nil represents an empty list

Another assumption made in representing programs is the use of compiler annotations in the compiled graph to efficiently represent recursive reentrant subgraphs in SPARO instead of using the fixed-point combinator Y [20]. (A compiler annotated recursive subgraph implies that the CG nodes constituting the recursive code will be specially marked so that it will be distinguishable from the remainder of the graph.) Compiler annotations will also be used to distinguish reentrant iterative code. More discussion on handling recursion and iteration in SPARO is provided later.

The primitive functions considered will include the primitive EQ for comparison and COND for conditional testing. The arithmetic primitives will consist of the unary MINUS and the binary PLUS operators. Only one Boolean primitive NOR (or NAND) will be considered. Separate input/output functions IN and OUT will be assumed to read and write data to and from SPARO. The input and output primitives are implemented as special messages sent or received on the network.

7.2.6 Graph Representation

The limited arity or out-degree of nodes in the CG makes direct mapping of the graph onto an array of processor very attractive. In such a representation, there are as many processors as there are nodes in the graph. Figure 7.4 shows how any arbitrary graph can be converted into a binary graph, which in turn can be represented with a two-dimensional (2-D) arrays of bits. Each CG node is represented by a fixed-size register. If each node is now provided more information about its location in the CG and some computing power (converting it into a processor), then CGR can be accomplished in a truly distributed fashion.

To describe the node in a binary graph such as the CG, three essential fields are required: the label or address of the current node, and the addresses of the LC and RC. Nodes with arity higher than two can also be represented as a binary graph. Since each node is of fixed size, any CG can be now represented by a 2-D array of bits. This is significant, since a 2-D plane of binary data can be operated on in parallel in optics. By this representation format, any program can be compiled into a CG, mapped into a 2-D representation, and then reduced or evaluated optically in one plane.

The linear representation of a node in the CG described above is not sufficient for executing CGR. Other information is necessary for implementing instruction passing. Figure 7.5 depicts the other information fields required in each processor node of SPARO. Besides the child node information, the node also has to contain information on its state (evaluated or unevaluated, etc.), whether it is free or allocated, and information on its parent nodes. This representation of the CG makes graph traversal from the parent to child nodes easy, since information on the child nodes is included in data stored in each node. Graph traversal from a node to its parents is more difficult since a node can have multiple parents. Information on the parent nodes is important since the result of a subgraph reduction is always sent to its parent node. During reduction, the node representing the result of the reduction cannot be freed or deallocated from the array unless one can ensure that no other node is waiting on it, or the parent nodes. Since a subgraph can be shared by any number of nodes, a mechanism must be created so that all parents can be accessed from the root node of the subgraph. This would allow the result of the evaluated subgraph to be communicated back to all its parents. We examine how multiple parents are represented.

While a CG node has more than two parent nodes, they are connected by using a special '& node' that is used to create a binary tree of any number of parent nodes as shown in Figure 7.4. Two parent fields are chosen for the following reasons. First, we cannot allow a node to have an unbounded number of parents. The use of & nodes allows the effective number of parents to be very large without resorting to stacks or queues. Second, a single parent field cannot save an evaluation request from a second parent if the node is evaluating upon the request of the first. Third, if more than two parents are possible, the &-node can be used effectively to store two parent fields at a time. When more than two evaluating requests arrive from the parent nodes, they can be saved in the & nodes. This means that a single node is never burdened with sending messages to many parents about the results of its reduction. Another advantage is that messages can be passed faster through the tree of & nodes than if one node passes the message sequentially. On the other hand, some penalty is paid because one may end up with a structure resembling a linked list rather than an inverted tree, which would slow the delivery of results.

7.2.7 Combinator Graph Node Types

Since details on other aspects of graph representation were provided earlier, we list here only the possible data types that have to be distinguished during CGR. The use of the different data types will become clear in the context of how combinator reduction is accomplished in SPARO.

For simplicity, we will discuss graphs with atomic node values. Complex data structures, such as lists, will be represented as a combinator expression of list atoms and the CONS operator as described earlier. The data fields in a node of a CG are distinguished by the type of node it represents. Although it is not shown in the example CGs shown in Figures 2 and 3, the atomic values of variables, combinators and primitive functions are contained in leaf nodes. A leaf node is a node whose LC contains a value and whose RC is empty. The & node described earlier is used to string together multiple, more than two, parents of an argument. A node that is neither a leaf node nor an &-node is called a regular node. Thus, a node is of three possible types: regular, &, or leaf.

Each node of a CG thus has type information associated with it and its LC and RC fields as shown in Figure 7.5. The parent fields are always pointers and therefore do not require typing.

7.2.8 Single SPARO Node Execution

We now consider how a processor node in SPARO would operate as an abstract machine. This level of abstraction explains the basic operations of a processor when performing a reduction.

7.2.9 Abstract Finite State Machine Model of a Processor Node

Although each processor node is physically a finite state machine (FSM) that executes the different control sequences for reducing combinators and functions, it can be considered as an abstract FSM at a higher level. This level of abstraction corresponds to specifying the state of evaluation of the node. Specifically, a processor node can be viewed to be in one of three external states: not evaluated (nev), evaluating (evg), and evaluated (evd). Initially, all nodes are in the nev state. When the graph embedded in SPARO is to be reduced, the root node is sent an evaluation request. This request is manifested as an APPLY instruction from the root node and is sent to its leftmost descendant as prescribed by normal order graph reduction. The state transition of a node is determined by the external state information (nev, evg, or evd), the type, (leaf or non-leaf) of its LC, and the instruction it receives as input. Thus, the actual state of a node is determined by its external state and the type of its LC, while the input to FSM is the instruction received. Below, we discuss some state transitions that can occur.

If a node does not specify a combinator or a function, then the following actions are taken. A message is first sent to its LC to transmit the APPLY instruction. The current node is then placed into the evg state. If the node is a combinator or function node, then the associated macrolevel sequence is invoked. In the case of a function evaluation, the collection of all required arguments is initiated. This sequence of events was illustrated in the reduction example of Figure 7.2.

A transition from the evg state to the evd state is made if the node is reduced to a value. A node can be reduced to a value only as a result of a K or I combinator reduction or by a primitive function evaluation. After the reduction completes, a node sends an Eval-complete message to its parent. An evaluated node can then be deallocated only if no other parent is waiting on its value. When multiple APPLY requests arrive at a node with multiple parents, the requests are saved in the parent & nodes. Each &-node can hold two evaluation requests. The parent and & nodes are set to the evg state. When the child node is finally evaluated, the Eval-complete message is sent to all waiting parent nodes.

Note that,since subgraphs are shared and not replicated as in string reduction, whenever an evaluation is requested of a shared subgraph, new nodes must be created to represent the mutated subgraph. (This was evident in the S-reduction of Figure 7.3.) Thus, nodes have to be allocated during a reduction. Our experience in writing the macroalgorithms reveals that the complexity of SPARO is not in the actual reduction process but in the management of the shared subgraphs.

7.2.10 SPARO Computer Architecture

The SPARO architecture was designed so that many simple nodes can store a small amount of information and use instruction passing to perform CGR. The key guideline in the design was to reduce the complexity of symbolic processing in an arcitecture comprising of simple processors. Functionally, SPARO consists of three parts: a processor array (PA), a set of control elements called the network processor (NP), and the network. The PA, described in the previous section, is responsible for executing the instructions that perform CGR. The network is the intercommunication network required to provide the means of transferring messages between processors in the PA. The NP is the buffer and network access control mechanism that is necessary for handling and transferring message packets between the PA and the network.

The network of SPARO is different from typical interconnection networks only in that it has to transfer short messages with low delay times. It must, therefore, be of a sufficiently simple structure to minimize the message passing overhead.

The NP arbitrates the transfer of messages between nodes and assures that two messages cannot simultaneously reach a processor. The need for the NP can be demonstrated by examining what happens when a message arrives at an executing node. Since there is no stored program or program counter in the node, the instruction register and the working registers constitute its current state. If an instruction were to be passed to this node, it would overwrite the currently executing instruction leading to the loss of any partial results (possibly irreversibly). By buffering messages and only sending them into the processor node when it is idle, the NP assures the safe evaluation in the PA.

Another function performed by the NP that greatly simplifies the operation of the PA is the network access by messages, which is best explained by examining the network model assumed.

The fine-grained processing in SPARO necessitates a large number of processors in the PA. Since a message can be destined to any arbitrary node, due to the mutative nature of CGR, any node can communicate with any other node. This implies that the ideal network provides complete, preferably non-blocking connectivity. Thus, if there are N nodes in the PA, the complexity of the network is at worst of order N^2, as in a crossbar. However, the implementation and the control of large optical crossbars is not currently feasible. Thus, a network with limited connectivity must be :sed. The network chosen in the original design was one that was the most naturally compatible with the architecture and layout of SPARO - a synchronous ring network constructed from shift registers. The ring network can be best described as a circular chain of shift registers (see Figure 7.6): each shift register is associated with a node in PA. The size of each shift register is the same as the size of a message. The ring network operates much like an escalator between levels representing nodes. A message in a node can access the network if the corresponding slot in the NP is empty, otherwise the message has to wait. It is the function of the NP to continuously check the status of the network and off-load the message onto the network when an empty slot is available. By shifting the burden of accessing the network onto the NP, the nodes in the PA are free to carry on their computation. In Figure 7.6, the NP has not been shown explicitly but can be viewed as an extension of the PA, comprising special registers with separate control.

We have examined two network configurations based on the ring network for SPARO: simple rings and chordal rings, where the each register in the ring is connected to a distant node besides its immediate neighbor. Both of these are relatively simple, and have the regular connectivity required to map them onto a 2-D optical array.

While we will cover the details on the optical implementation later, the control structures in the PA and NP deserve more attention. These control mechanisms are described next.

7.2.11 Control Structures for CGR in SPARO

The basic node structure explained earlier assumes that the processor node executes some set of instructions using specific control fields and flags. To expose the implementation details that must be considered, as well as illustrate the level of the complexity of each node, we list the key ones below.

Our preliminary analysis of combinator reduction and functional evaluation algorithms reveal that five Boolean flags are necessary for use by the macrofunctions or macroinstructions in each node. These are:

- 2 parents present : This flag indicates that both parent fields of the node are occupied, or that the node has two or more parents.

- 2nd apply received: This flag indicates whether the second parent of a node has requested an evaluation. It applies only to nodes for which 2_parents_present is true.

- doing S: This flag indicates that at the top level (root node of a combinator subgraph) an S rather than a B combinator is being executed.

- awaiting arity incr replies: This flag indicates that two new parents have been added to the parent list of a shared argument node (in the case of a S, B, and C combinator reduction).

- awaiting eval completes: This flag indicates that both arguments of a binary function have been evaluated; the binary function can now be applied.

Further, two other fields are used to describe a node type and the level of the combinator subgraph. These are:

- node type: regular, &, and leaf

- combinator level: 1, 2, or 3, where the number refers to level in the combinator subgraph; 1 is the level of a node whose LC is a combinator (S, B, or C) and RC is the first argument, 2 is the level of a node whose RC is the second argument, while 3 is the level of the root of the combinator subgraph.

We have assumed that leaf nodes for atomic values in the CG, i.e., atomic values are separate nodes. While this results in inefficient use of storage, it implies a more efficient control. Maintaining separate identities for atomic values implies that every argument is a pointer to a value. Such a representation eliminates the need for checking whether every argument is a pointer or value, and considerably simplifies the macroinstructions (macros) executed in each node.

In addition to the aforementioned flags, an annot flag is used to indicate that a node is annotated and cannot be reduced on first evaluation. Annot nodes are used to describe recursive and iterative code, and are dealt with later.

Memory management, such as node allocation and deallocation, is crucial in all symbolic processing, especially in the fine-grained computing in SPARO. This is because the node space can be quickly exhausted even by small recursive programs if garbage collection is not employed. In the case of SPARO, the memory management functions are directly built into the control algorithms to explicitly control deallocation and allocation of nodes. A separate state bit dealloc is used to denote if the node is free or in use.

7.2.12 Macroinstructions and Messages

The communication between the PA nodes is handled by message macros. Within a macro, a message (denoted by the instruction SEND) is specified by four fields: destination, instruction, data1, and data2. The source field is not specified since a message initiated in any node will always use the current node address as the source. Functional units that execute arithmetic or logic primitives, such as PLUS and AND, are assumed to be located by a message with the destination address set to the name of the function (denoted by functional_unit). This assumption allows the implementation of the control unit of SPARO to be independent of the implementation of the functional units.

To explain how messages are incorporated into the macroinstruction (macro) executed by a processor node, we describe as an example the macro that would be used to execute an APPLY instruction when the node is in the nev (not evaluated) state. A somewhat different and much simpler macro is executed if the APPLY is received when the node is in the evg or the evd state. In_buffer and out_buffer refer to registers in the node that receive and dispatch messages from and to the network, respectively. P1 and P2 refer to the addresses in the first and second parent fields of the node. The instructions in the macro are explained in the next subsection.

APPLY MACRO FOR UNEVALUATED NODES

```
    LOAD       P1 <- in_buffer.destination /* load the parent
               address */

    CASE       node_type of: /* take different actions for
               different types of nodes */
```

```
                 regular node: /* send a message down to the
                 left-child node */

        SEND
                 out_buffer.destination <- LC
                 out_buffer.instruction <- APPLY

        LOAD     state <- evg
                 &-node:        /* send a message down to the
                 right-child node */

        SEND
                 out_buffer.destination <- RC
                 out_buffer.instruction <- APPLY

        LOAD     state <- evg
                 leaf node:     /* start evaluating by sending data
                 to parent node */

        SEND
                 out_buffer.destination <- P1
                 out_buffer.instruction <- Eval_complete
                 out_buffer.data1 <- RC

        IF NOT   (2_parents_present) then     /* only one parent -
                 deallocate */

        LOAD     state <- dealloc

        ELSE          /* two parent nodes exist - wait for 2nd
                 APPLY */

                 IF 2nd_apply_received then

                      SEND
                           out_buffer.destination <- P2
                           out_buffer.instruction <- Eval_complete
                           out_buffer.data1<- RC

                      LOAD state <- dealloc

                 ELSE
                      LOAD state <- evd /* wait for 2nd APPLY */
```

7.2.13 Basic Control Instructions

Having described how messages are constructed, we provide a summary on the construction of macros executed in the nodes. A basic set of control instructions or mini-instructions is used to control the operations within a processor node, the sequencing of instructions, and the processor memory management, as the APPLY macro example above illustrates. These instructions are described in an assembly language format.

Three basic mini-instructions are used to move data between registers, initiate data transfers between processing nodes, and control basic and macro and mini-instruction execution.

The data transfer instruction is LOAD, which moves data from register to register within a node. Condition flags are also set with the LOAD instruction.

Conditional statements are represented by IF-THEN-ELSE structures, while CASE instructions are used to represent multiple IF-THEN-ELSE sequences.

The SEND instruction is used to construct messages in the processor node to be dispatched to the network by the NP. It sets up the data in the output buffer (in the NP). This buffer contains the four message fields, destination instruction, data1, and data2. The network processor then sends a message containing these four fields and one indicating the current node (source).

7.2.15 Control Requirements in the Network Processor

While the use of the network processor combined with the limited number of parents and children solves the contention problem for high levels of parallel data movement, we have yet to address the lateral data moves required in SPARO. These data moves are critical when considering the implementation in optics.

We list here only the lateral data moves that must be realized in SPARO. For implementation, one would select a minimal number of possible data field moves in each of these categories. These are:

- Network to NP to fields,
- NP to PA node fields,
- PA node to NP fields, and
- NP to network fields.

Note that because the NP works in concert with the processor and the network, it will require its own set of "instruction" registers which would keep track of its state relative to that of the processor and the network. These instructions will control and sequence the moves to the network and the PA nodes. As with the PA, SS will be employed to set up the sequencing and gated interconnects will provide the actual data movements.

7.2.15 Memory Management in SPARO

The memory management operations are of two types: allocate and deallocate. An allocate operation is used to create and assign new nodes in the processor, while a deallocate instruction is used to free nodes that are not required anymore. The deallocate instruction is thus used to perform garbage collection. Since we are limited in the size of the array, memory management is an important function in the processor.

7.2.15.1 Allocate

This function uses the network to allocate a node in the graph. A message to a free node is sent using an indefinite form of the SEND instruction as shown below:

```
SEND
     out_buffer.destination -> ?
     out_buffer.instruction -> instr
     out_buffer.data1 -> data1
     out_buffer.data2 -> data2
```

where ? indicates any available node in the PA, and instr, data1, and data2 refer to an instruction and data to be passed to the newly allocated node. The network delivers this message to any node that does not have its alloc bit set.

7.2.15.2 Deallocate

The complement of the allocate operation is the deallocate operation. It is more complicated than the allocate operation. This is because a node cannot be deallocated if it has any parent node waiting for its evaluated value. As previously described, a node can be deallocated only when no parents are waiting on it.

The basic operation of the deallocate operation is to check if the node has two parents. This is done by examining the flag 2_parents_present. If this flag is not set, then the node has only one parent and can be deallocated. Otherwise, the node checks if it received an evaluation request from the other parent. If it did, then it sends its evaluated result and then deallocates itself. If the evaluation request has not been received from the second parent, the node cannot be deallocated and must wait. Checking for multiple parents is explicit within the macro that invokes the deallocate operation. This is evident in the deallocation of the leaf node in the APPLY macro example described earlier.

7.2.16 Functional Units in SPARO

One of the components currently missing from SPARO is the set of functional units. For SPARO to be complete it must be able to perform more than just the functions required for combinator graph reduction. It needs to have primitive arithmetic and logic functions included in it. Two approaches exist for completing SPARO. The first would be to add a functional unit to each node. The second would be to have special nodes attached to the network for the sole purpose of executing the primitive functions. We describe the tradeoffs for each approach.

Assigning a functional unit to each processor would provide the best performance since it avoids a large number of data movements and message delay overheads. However, since most nodes do not, and may never, perform the primitive functions during the execution of symbolic processing, the addition of functional units is very expensive in terms of unused computational power. (The situation would be different if SPARO is used purely for numeric processing.) The addition of functional units to each node would also require a larger optical system and a correspondingly longer cycle time. This waste of computational power and longer cycle time may only be justified if it leads to substantially better system performance than the alternative.

The second choice for implementation of functional units would be to use special nodes on the network for performing the primitive functions. This approach would greatly simplify the nodes and would not add a large amount of seldom-used hardware to the system. However, such a design would increase the message traffic on the network and thus the time required for function evaluation. Also, the many requests for a few resources would result in contention problems, which would be addressed by buffering requests. The buffer location and mechanisms need to be defined with careful regard to optics capabilities and limitations.

In summary, the design of incorporating functional units in SPARO is governed primarily by the physical and technological limitations, and can be decided only at the time of implementation.

7.2.17 Complex Control Structures: Recursion and Iteration

A general-purpose symbolic processing computer must allow the execution of high-level programming constructs such as recursion and iteration. While simple function evaluation is not difficult to define in SPARO, the implementation recursion handling needs some explanation. Here we show how a recursive and an iterative combinator graph can be evaluated in the fine-grained SPARO architecture. With the inclusion of these facilities to execute different programming constructs, the SPARO architecture becomes a very versatile evaluation engine that can execute any program compiled into a CG.

7.2.18 Recursion

The typical method for executing recursive code is to expand out the recursive call until closure is reached. The control flow sequence is usually maintained by using a stack. Since stacks and other explicit memory structures are difficult to implement in optics, we have chosen an alternate route. Instead of executing shared code and saving different contexts on stack, we copy the subgraph for the recursive structure and evaluate or reduce each instance separately. Each instance of the recursive code is explicitly generated and reduced during CGR. There are two implications to this approach. First, the recursive subgraph has to be copied each time. Second, the original subgraph cannot be reduced, and thereby destroyed, before closure is reached. These two issues are examined next.

Employing the first approach is very expensive since the complete subgraph is first copied at runtime and then executed. However, such a 'copy and reduce' technique may not be necessary. One possibility is to pipeline the copying and reduction operations. Thus, while the original graph is being copied (in normal order), the apply instruction can be sent down the incomplete graph in normal order. Since copying requires allocating free nodes and sending data via messages, the speed of copying will depend on the speed of the network. To prevent evaluation of an incompletely copied node, certain control constructs are required to suspend evaluation until its child nodes have been specified.

Normal CGR reduces and thus destroys the original CG. That is, the CG is not reentrant. Since we do not have separate memory, we must make other provisions for maintaining the recursive subgraph. To avoid use of a separate special memory, we ensure that the original recursive graph is not destroyed until closure is reached. To distinguish the recursive portions of the code from the rest during execution, we use compiler annotations for all nodes that constitute the recursive subgraph. An annotated node will have its annot bit set. Note that another option to the annotated node technique would have been to distinguish between compile-time generated nodes and runtime generated nodes where the compile-time generated nodes are never destroyed. Conceptually, there is no difference between the two options, and so we have decided to examine the annotated case.

When the root node of a recursive subgraph receives an APPLY instruction, it initiates a copying algorithm. If an incremental copying algorithm is used, that is, copying and graph reduction are pipelined, it can be executed recursively. A COPY macro does the following: the present node is first copied in a newly allocated node (the allocation is also initiated by the copy instruction), and a copy instruction is sent to its LC and RC. Each child node then invokes the copying algorithm again. The copying stops when leaf nodes are reached. After the first node has been copied, an APPLY instruction is sent to the first copy of the recursive subgraph. In this manner, the copying and evaluation of the subgraph can be accomplished in a pipelined fashion.

The correct sequencing of the recursive evaluation is specified by the linking of the copies of the subgraph. Let the head or root node of the original recursive subgraph be called H. The node that specifies the recursive call, i.e., the node in the body of the subgraph that points back to H, will be denoted by T for tail. These nodes in the copies will be called H_1, T_1, H_2, T_2, etc. The correct sequencing for evaluating the recursion proceeds as follows. The parent node P of (H,..,T) sends down a request for evaluation, an APPLY instruction, to H. Since H is the first annotated node, it initiates the first copying sequence. The copy consists of the sequence $(H_1,..,T_1)$. The tail T_1 is set to point to the original head node H. H sends an evaluation request to H_1 with the address of its parent P. Since the subgraph $(H_1,..,T_1)$ is not annotated, it will be reduced. If the recursive condition (the condition that determines the recursive call) is satisfied, T_1 is evaluated. Since T_1 points to H, another copy of the original subgraph is made with the new tail T_2 pointing to H. H_2 is then sent an evaluation request from H with the address of T_1, since T_1 is the new parent of H. The subgraph copies are thus linked to emulate a nested call structure.

The recursive spawning of new copies continues until closure is reached, when the recursive condition is false, say at the kth copy. At this stage T_k is sent a deallocate message since the recursive condition is false. Since T_k points to the original subgraph, (H,..,T) is recursively deallocated. In parallel, the kth copy reduces and sends back a completion message to H_k, which in turn sends a completion to T_{k-1}, the tail of the (k - 1)th copy. The (k - 1)th copy of the subgraph is then reduced. The reduction proceeds until the first copy is reached. Since P is the original requesting nodes, the final completion message is sent from H_1 to P.

It would appear that handling recursive calls is more efficient if block copying, that is, copying a complete sequence $(H_i,.. T_i)$, rather than incremental copying is used. This does not affect the control sequence described above. In block copying, a complete block of nodes that constitute the recursive subgraph can be copied in a single step. This avoids the problems of flooding the network with messages to allocate nodes and to check if child nodes have been allocated. Block copying can only be feasible if the size of the recursive subgraph is relatively small.

7.2.19 Iteration

The problem of maintaining the iterative subgraph until iteration is complete is similar to that of the recursive case. Therefore, a similar solution using compiler annotations is chosen. The only difference is that, unlike in recursion, the life of the iterative structure is usually known at compile time. Furthermore, the different iterations do not exist at the same time. Only the current iteration must be saved. The copying of the original subgraph is initiated if the termination condition is not satisfied. A reduction can proceed as soon as the copying has completed or is in progress in the case of incremental copying. Each iteration is completely reduced before the next iteration is generated since no nesting of subgraphs occurs.

Having examined the detailed evaluation strategy for CGR in SPARO, we now examine the optical implementation of the different components of the architecture in the next section.

8. Optical Implementation of Sparo

This section outlines possible implementations of each of the component systems that make up SPARO. We first describe the basic optical components assumed available and employed in SPARO.

8.1 Optical Primitives

Only a few optical elements are required to implement the structures necessary in SPARO. It is assumed that suitable nonlinear optical gates can be found to perform the logic operations required. Interconnects between logic planes will likely employ classical optical components such as mirrors, beam splitters, and lenses as well as holographic deflectors. Much work is currently being done in these areas [40,41], and we believe that they will ultimately provide the required components and subsystems.

The two techniques that SPARO employs are Symbolic Substitution (SS) [41] and gateable interconnects. SS is a technique to perform complex logical operations by the manipulation of patterns. Its main attraction is that it is well-suited to operation in parallel and it has a relatively straightforward implementation in optics. Gateable interconnects are nothing more than masks that allow light to be transferred to specific locations in the next logic plane in the system. In their simplest implementation, they might consist of optical gates which control light through a subsequent classical interconnect. In SPARO, SS is employed to perform the control operations within a node, and gateable interconnects are used to transfer information between data fields.

An optical system that performs one SS rule consists of two portions: the recognition optics and the scribing optics. In the recognition optics, the fields where SS is to be performed are optically split into several copies. These copies are then shifted in varying directions and distance and imaged onto a plane of optical gates operated as thresholding elements. In this manner, specific patterns can be recognized. The light out of the optical gates denotes locations where a pattern was recognized. In the scribing optics, this output light is optically split into several portions, then shifted and recombined creating the desired output pattern.

Each rule in a SS system requires a specific recognition and scribing optics for that rule. It may be possible, however, to combine shifted images, or partial results of recognition and scribing to reduce the complexity of many-rule SS systems.

Gatable interconnects could be realized using a pixel from a control field to control several optical gates configured as a

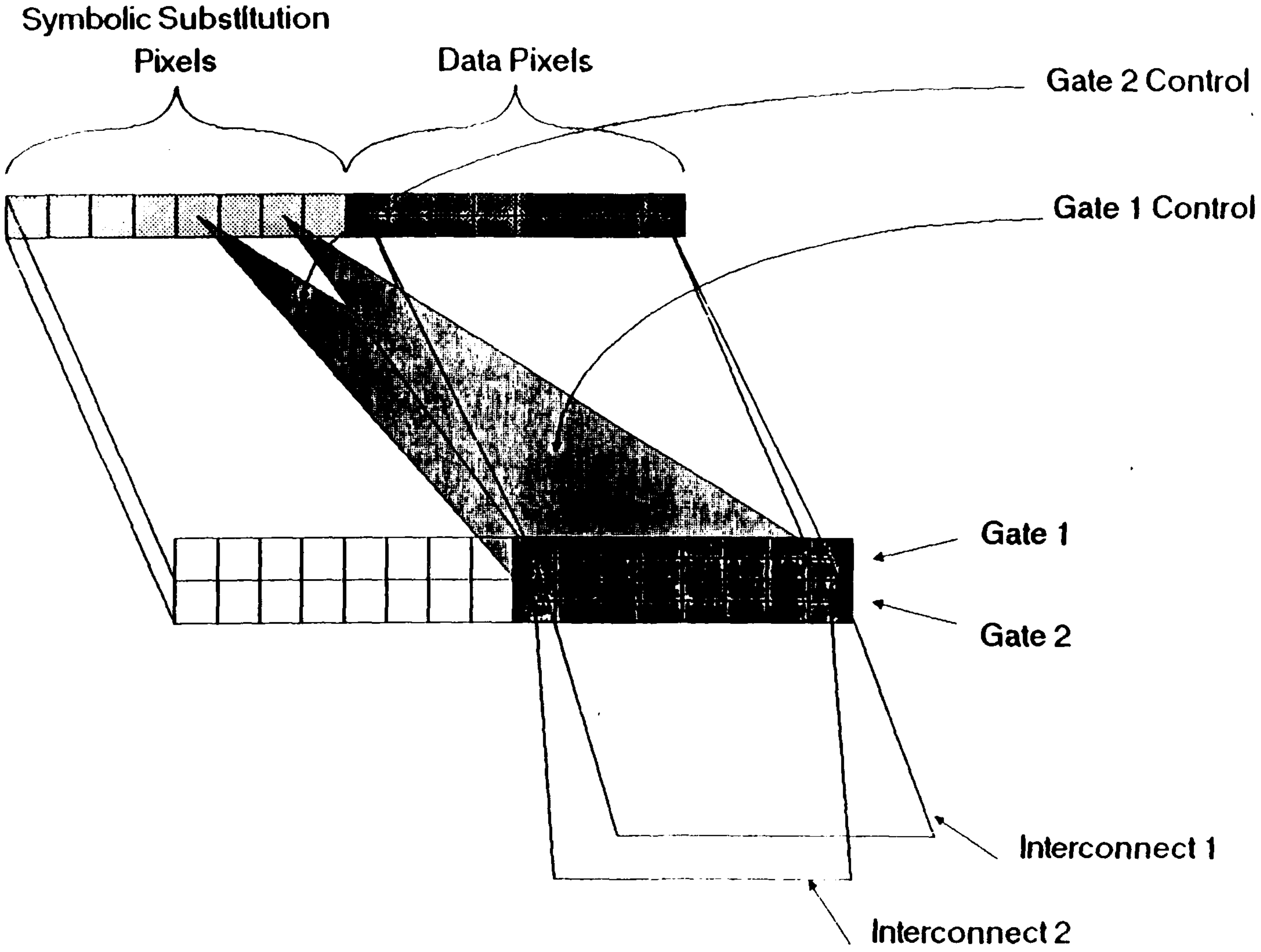


Figure 8.1 Moving data with symbolic substitution and gatable interconnects

programmable mask. If the light from the control pixel is present then light would be passed through the mask. Gateable interconnects could be realized by using such programmable masks in the paths of light connecting the output from data fields to the input of other data fields. Figure 8.1 shows how SS and gateable interconnects could be used to move data around in an optical system. Specific pixels in the region where SS is performed are used to control masks that send information from one register A to other registers B and C. This use of large fanout is contrary to conventional thinking about the use of optical logic gates, but it avoids the need for complicated fanout circuitry. It is clear that the use of gate level optical computing strategies [41] would allow these functions to be implemented without the need for fanout. However, it is likely that less than regular interconnects will be required at the lowest levels of the system.

8.2 Optical Layout

As described earlier, the optical system is laid out as a linear array of nodes (Figure 8.2). The elements of this plane are optical gates that provide the nonlinear functions required for SS (Figure 8.3). Free-space optical interconnects provide the connections between fields in the nodes and on the network as well as for SS. Connections are provided for all processing elements in parallel to facilitate simultaneous operation of all of the processors. The linear layout is required to reduce the number of possible interconnects between pixels on the plane so that diffraction effects can be managed.

Figure 8.2 shows schematically how the plane is functionally split into three sections and how that plane makes up part of a larger system. The block in the lower portion of the figure represents the optics employed for the interconnects between planes of nonlinear elements. Logical operations are performed by interconnecting the outputs of optical gates from one plane to the inputs of gates in the next plane. The blocks and planes correspond to the blocks and planes in Figure 8.3. A complete cycle of light around the loop shown in Figure 8.3 represents one machine cycle. The various stages are required to perform the operations necessary for SS and to set up the transfers between registers.

To explain the macro-architecture we will examine the interaction between the processor portion of the optical plane with the network portion. An intermediate stage is necessary to handle the management of data transfer between the network and PA, the processor array. We have referred to this stage as the NP or the network processor.

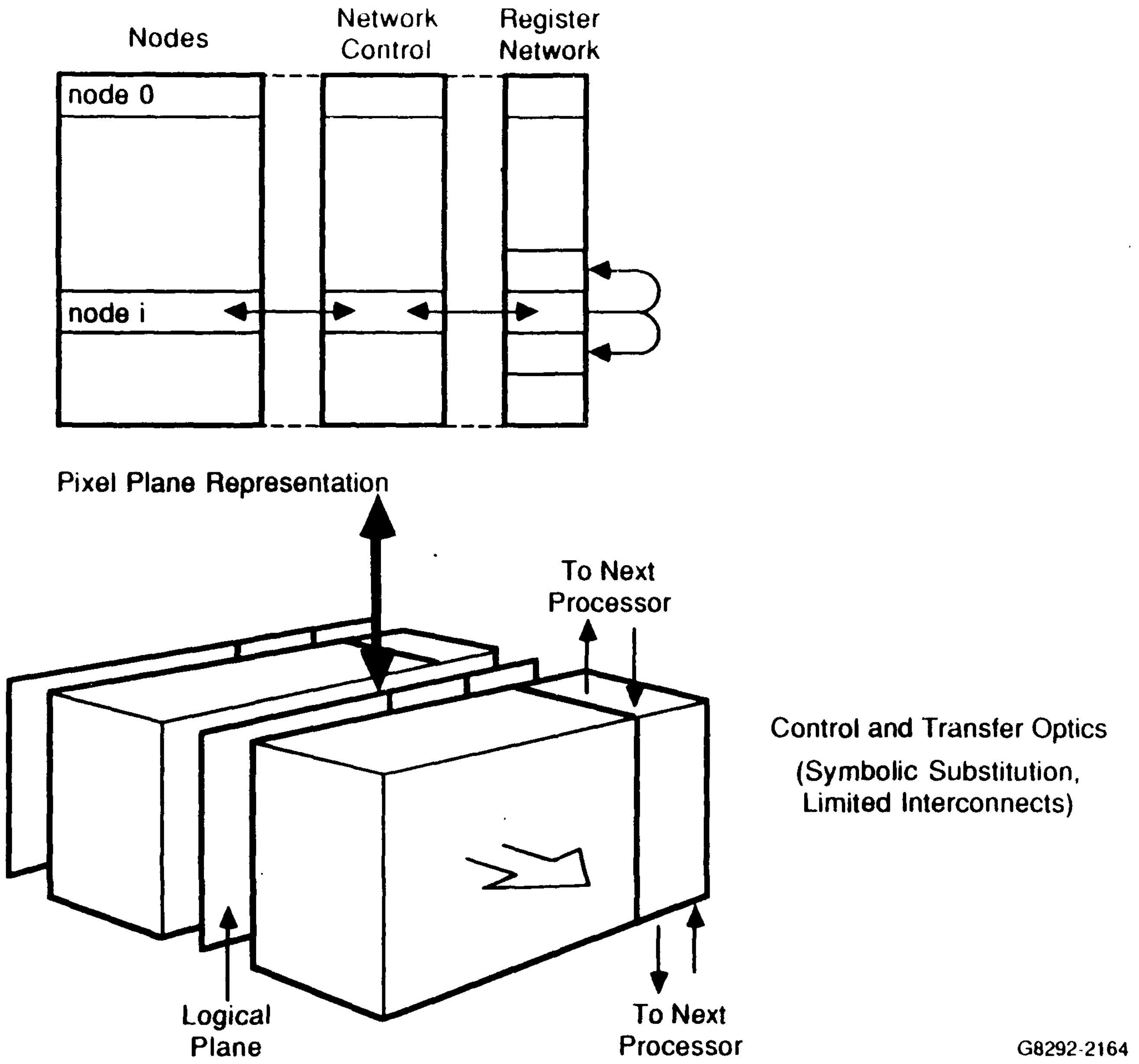


Figure 8.2 Splitting plane into three sections

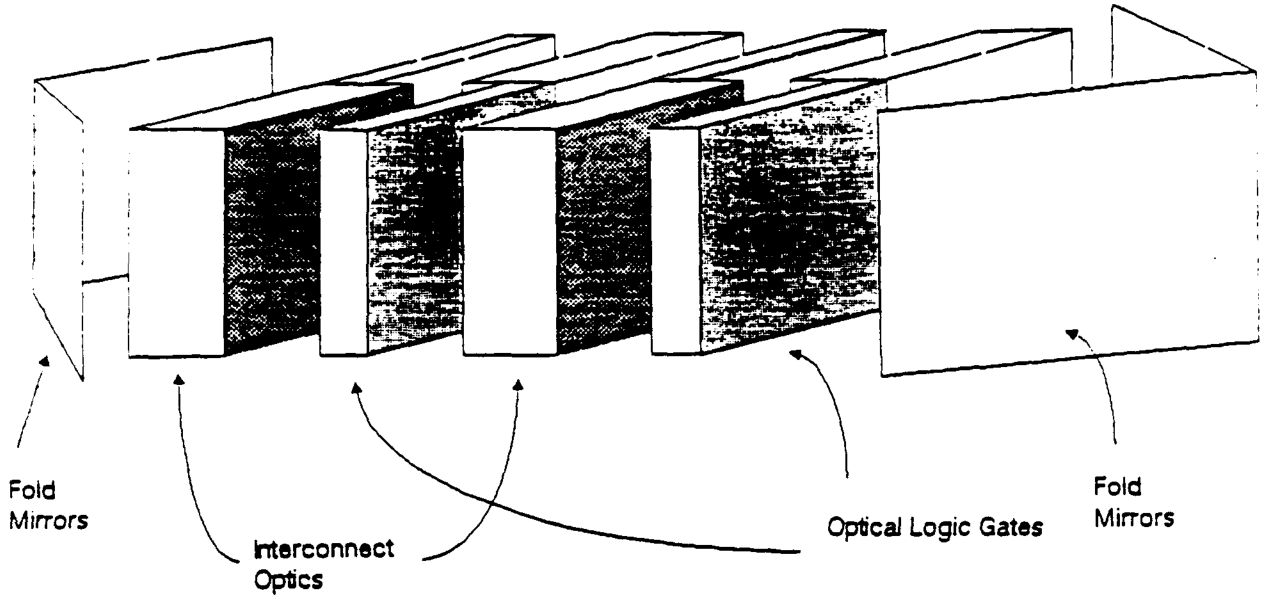


Figure 8.3 Blocks and planes of the optical system

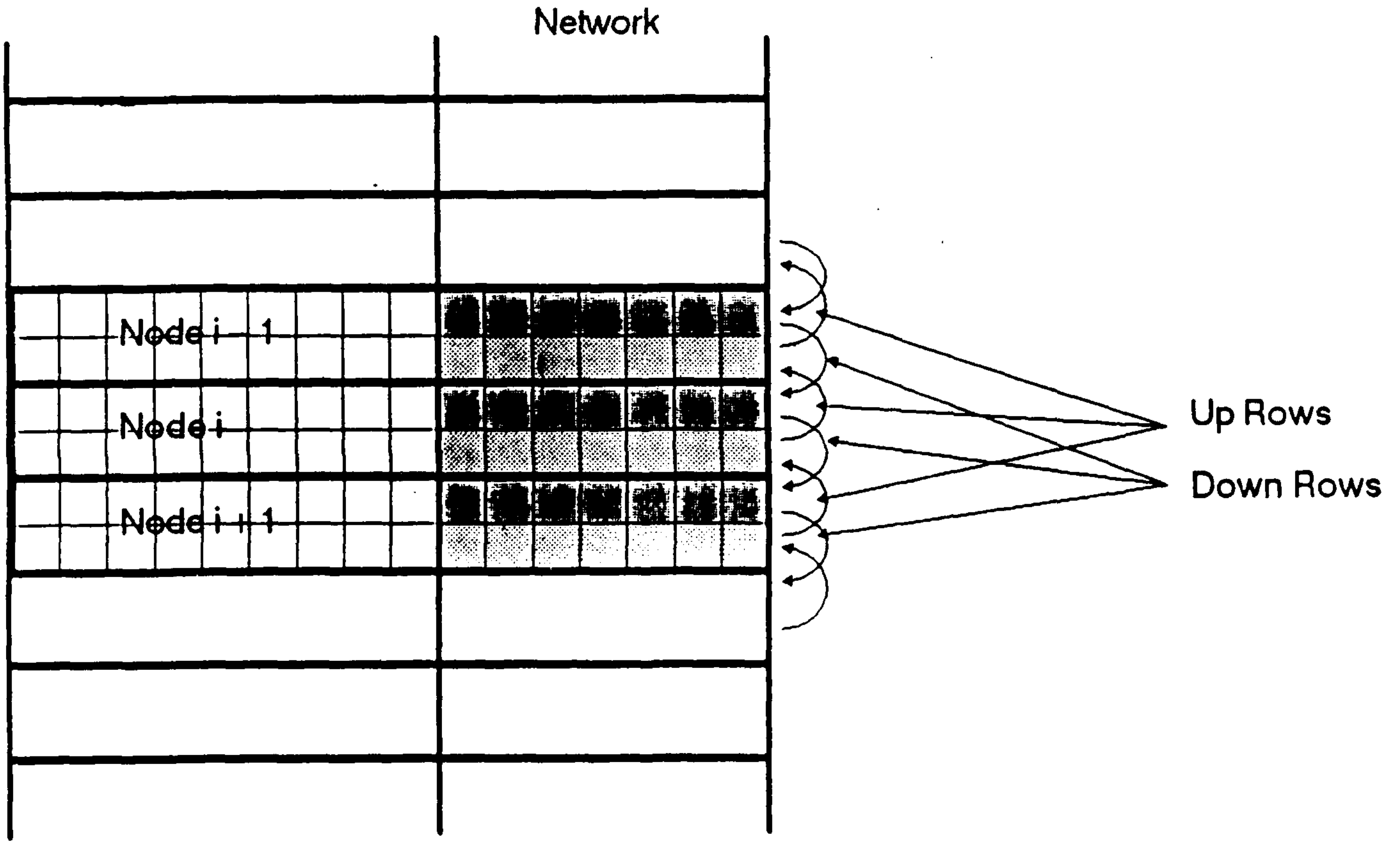


Figure 8.4 Network format for implementing bidirectional searches

8.3 The Architecture of the Network

The network used by SPARO is best described as a micro-area network. It has an extremely simple protocol, but it can pass messages in parallel between the many nodes of SPARO. As was evident from the macro-level descriptions of the SKI combinators, it is used by the processor by invoking send instructions. Here we examine one possible configuration of the network. It was not designed for high performance, but rather to demonstrate that a communications channel could be developed that was in keeping with the overall system requirements.

The requirements used to develop this network, and which must be met by any alternative, are:

- parallel or near-parallel transmission of many messages,
- simple (from the node processor point of view) contention management,
- supports node allocation,
- nonblocking (partially managed through processor design),

The functionality of the network is determined by the nature of operations required for CGR. The gross functions that are required as capabilities of the network are graph traversal and data moves. These in turn require that the network must be able to locate nodes in the array by their node numbers, and be able to read/write data in and out of the data fields.

Another required feature of the network is the parallel access of nodes in the plane. This is critical since we want to exploit any parallelism available during CGR. When different subgraphs can be reduced simultaneously, the network should be able to accommodate the data moves and traversal in disjointed segments of the graph. Efficient parallel access of the nodes in the graph implies that there should be no contention in the use of the network. As we will see later, addressing contention will strongly influence the design of the architecture.

The basic operation of the network is to sequentially step a message from node to node along the linear array. If the destination matches the neighboring node, the data is transferred from the network to the network processor.

We will assume, in our implementation, that the network searches for the node number sequentially in the proper direction (up or down) in the array. The sequential search means that at every node a comparison is made between the destination node number and the current node number. While this is slow, the advantage is that both the processor and the network (and also network processor which manages contention) can operate at the same speed and with the same cycle time. Searches for nodes can be done in either direction to allow for parallel searches crossing the same node without contention. Such a bidirectional search is implemented by

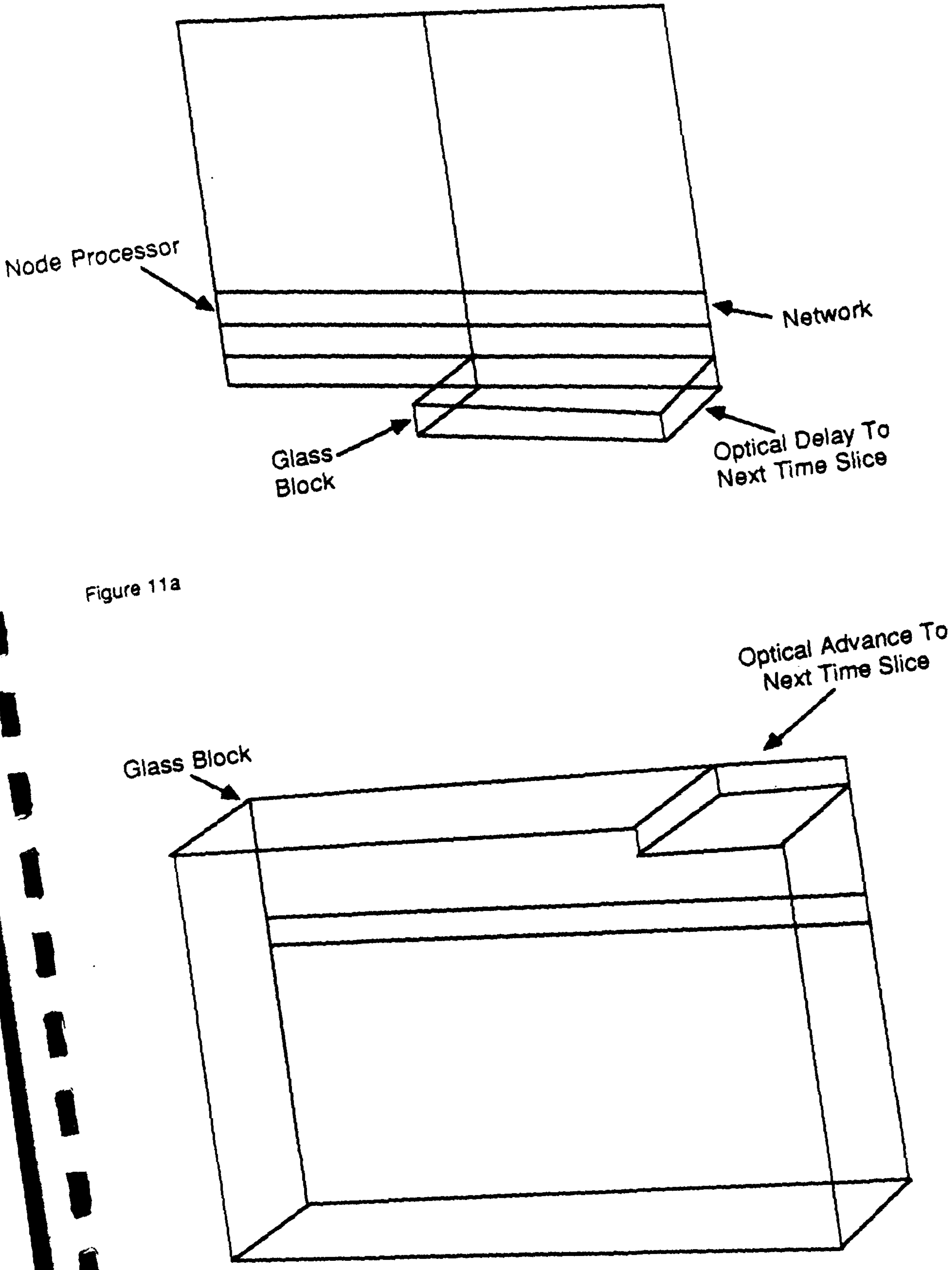

Node Processor
Network
Glass Block
Optical Delay To Next Time Slice
Optical Advance To Next Time Slice
Glass Block

Figure 11a

Figure 8.5 Optical time slice advances

providing two sets of fields for each node in the network. Figure 8.4 shows the format of the network for implementing bidirectional searches.

The network can be viewed as a sequence of registers, one set of which transfers data up the array while the other set transfers data down. In each register set, the basic operation is the transfer of one register to the next as in Figure 8.4. On each processor cycle, register data is moved up (or down) one step. Data transfer is initiated by placing a message on the network. To place a message on the network, the processor first determines the relative location (up or down) of the destination. It then checks to be sure that on the next machine cycle its network register will be free by examining an 'occupy' pixel or bit. If it will be, the processor transfers the message to the network. If not, it checks again for a freed register.

Connections to the edges of the processor plane could be made to other processing elements (to increase the total number of processing nodes in a system) or they could be refracted to the elements on the other end of the array as shown in Figure 8.6.

On every cycle the network compares the message destination to the current position of the message. This is accomplished by comparing the node number field to the destination number field of the message. If the message is at its destination, it is moved off the network into the network processor. Otherwise, it is passed to the next register in sequence.

As described, contention for the network is managed by refusing to send a message if the network is busy. This method of managing contention reduces the buffering requirements of the system since the node can be made to buffer the information while it is waiting to send it out. Further, this method does not introduce more inefficiencies in execution since a node, in most cases, does no processing until its subgraphs have been reduced. This means that it can idle while waiting for access to the network.

Since multiple processes can be occurring on the same graph, a message may arrive at a node that is still processing. In such a case, the message must be buffered until the current processing is complete. This buffering is provided by the network processor. The network processor needs to manage only two buffers for messages from the network. Two buffers are sufficient since each node has only two parent or child nodes. Such limited connectivity constrains the number of messages that can arrive simultaneously at a node, and thus limits the number of requests that may have to be buffered.

An approach to increase the performance of the network would be to implement a chordal ring that was mentioned earlier. The increased connectivity of the chordal ring would reduce the delivery time of a message by providing pathways that skip portions

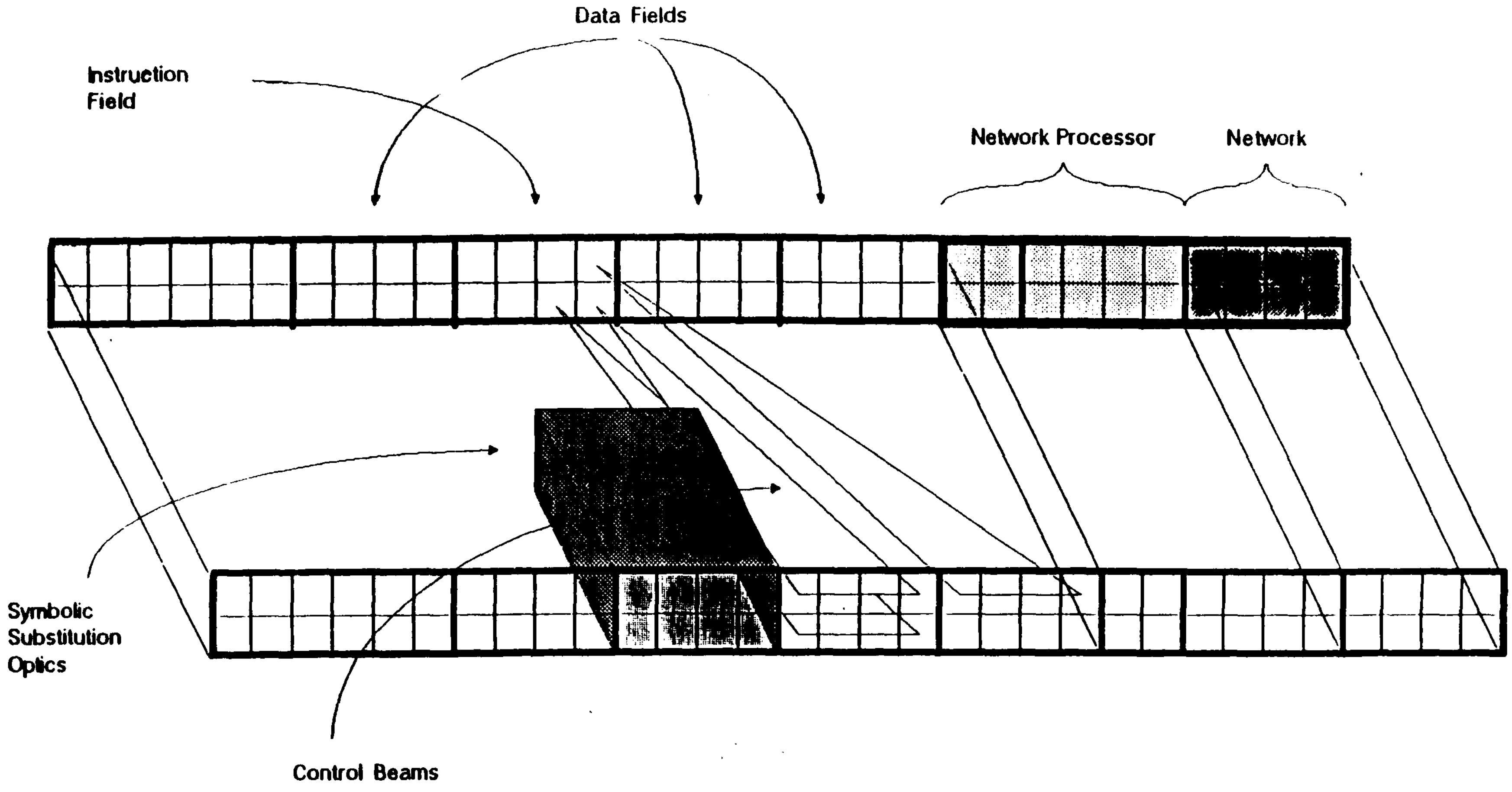


Figure 8.6 Basic processor functionality

SPARO processors. This is the performance available when the program executed is totally sequential. Second, we determined the total throughput when a parallel application is executed. Note that in theory, because of the nature of CGR, the parallelism in the application is limited mainly by the parallelism inherent in the application and the ordering imposed when using normal order graph reduction with some eager evaluation (for strict and semi-strict functions). There are no limitations of trying to map a parallel algorithm to a specific architecture.

The criterion used for measuring performance of graph reduction machines is reductions per second (rps). Special-purpose electronic graph reductions have been designed to execute at 100,000 rps to at most 300,000 rps [42,43]. Since different combinators and functions require different levels of computational effort, these figures represent an average behavior on applications used for benchmarking. Due to a lack of benchmarking tools for graph reduction machines, our approach to analyzing a theoretical architecture is based on estimating the performance of SPARO on reducing an 'average' combinator which we define below.

We will assume, and later compute, the cycle time of the SPARO machine. This corresponds to the time taken by the SS optics and the gated interconnects to execute a single mini-instruction. Since each combinator and primitive function is a sequence of macros that can be expressed in terms of mini-instructions, we can estimate the execution times of each basic combinator and functions that form the core of the instruction set for SPARO. An average of these cycle times for all functions (and combinators) will provide an estimate of the execution time of an 'average' reduction. The assumption is reasonable if variances in the cycle times for different functions are within one order of magnitude. This is indeed the case as our analysis of all macros revealed. The serial reduction rate, SRR, can then be defined simply as follows:

SRR = 1/(Machine cycles for average reduction * Cycle time)

The parallel reduction rate is difficult to estimate since this depends on the application. However, for a fine-grained message-passing architecture, where each new reduction is initiated by a message (APPLY), the peak parallelism in the architecture is limited by the peak throughput of the interconnection network used. In a first-order analysis, therefore, the parallel reduction rate can be define as:

PRR = SRR*(Throughput of network)

In the following subsections we present the results of our analysis on the estimates for SRR and PRR.

of the network. The interconnect pattern would no longer be local, but it would still remain regular and thus still potentially be implemented in optics. The connections of the nodes at the end of the processor could still be handled by the use of bulk optic deflectors as in the simple linear case.

A chordal network would operate much like the simple network, except the message would be passed through the far connection if the message was destined for that part of the network and the slot prior to the destination was empty (not about to be moved into the far slot). This would require a control system of greater complexity, but would still not require any extra buffering to manage contention. The added complexity comes in detecting the distance a message is going to travel. The simple compare for equality is no longer sufficient, but a subtract and detect sign will work and may be simple enough (relative to the complexity of the other operations) to warrant inclusion in the SPARO system.

8.4 Architecture of the Processor and Network Processor

Our discussion shows that the control operations are relatively simple and few, especially at the macrolevel. The aim for the development of an architecture for the processor is to determine the minimal set of instructions that need to be implemented.

The registers and functionality of the processor require that the processors store information and transfer it between fields; the same functionality is required of the network. However, in the case of the processor, the control for data transfers is provided by the executing instructions.

Currently, we have not completed the design of the processor architecture, but have schematically developed how it will operate. Figure 8.7 shows, schematically, the basic functionality of our concept for the processor. Bit fields in the instruction register control gateable interconnects between different lateral moves. The bit fields in the instruction register, in turn, are controlled by SS rules. We envision that the SS rules would be developed from the mini-instructions in a two-step process (mini-instructions to microcode-like instructions to SS rules) so that the number of SS rules can be minimized.

9. SPARO Performance Analysis

The feasibility of the SPARO architecture, however novel it may be, depends on its potential performance. Since this is an exploratory architecture, the performance analysis also establishes bottlenecks in the design. The performance analysis was done in two phases. First, we determined the serial performance of the

9.1 Serial Reduction Rate

A typical macro can be represented as a sequence of parallel loads (transfer from one register to another in the PA or NP) and parallel or serial sequence of messages. The machine cycles required for executing an average reduction is thus the sum of the cycle times taken to execute the loads and transfer the messages. It is expected that the delay in delivering messages is larger than the average time taken to transfer a message. This is because in a network of limited connectivity, multiple cycles are necessary to transfer a message. Thus, the SRR can be approximated by the delay of the average number of messages. A study of the combinator and functions reveal that the average number of messages required is 8.1, assuming that all combinators and functions are equally likely to appear in the CG. (This is not too unreasonable since an arbitrary CG can have almost any combination of combinators.) We note that the number 8.1 is an optimistic estimate, since in the worst case the arguments of a combinator or function can be shared and may be located at the end of a chain of & nodes. Traversing a sequence of & nodes will require more messages to reach the arguments. The SRR is therefore equal to the inverse of (8.1*Average Message Delay*Cycle Time).

In the case of the ring network of size N, the average message delay is N/2 for a unidirectional network and N/4 for bidirectional network. Clearly, the delay grows proportional to the size of the network or the size of the PA. If the cycle time is T, the average message delay is NT/2. Thus, the serial performance is inversely proportional to the size of the network. For a PA with 1K nodes, the serial performance will be three orders of magnitude slower than the cycle rate for the basic optical system. Clearly, the ring network is the bottleneck.

One option considered was to use chordal networks where every node in the ring is connected to its adjacent neighbors as well as to nodes that are a distance k (k > 1) away. The optimal performance is obtained when k = N. However, with this increased connectivity, more contention checking operations are required. If three cycles are assumed to be sufficient to accomplish contention checking, the average message delay is 3N cycles for a unidirectional chordal ring, and 3N/2 for a bidirectional chordal ring. For a PA with 1K nodes, this yields a SRR that is two orders of magnitude slower than the cycle rate of the basic optical system. Thus, the use of the chordal ring improves the serial performance by an order of magnitude.

9.2 Cycle Time of SPARO

To estimate the performance of SPARO, it is necessary to determine how long a machine cycle will take. We define a machine cycle as the time required for light to traverse the length of the processor and return to the starting point. Since many logical operations are required in each machine cycle and many optical interconnect paths are used, the cycle time is determined both by the speed of light and by device speeds. However, to set an upper bound on the performance, we assume that the device response times are infinitesimal; the devices are infinitely fast. Similarly, we do not have any quantitative description of the complexity of the optical interconnects to enable the exact calculation of cycle times. However, we can estimate the performance based on an estimate of the number and complexity of SS rules required and the size of the optical system required to implement that number of rules. We will also, for the current analysis, ignore the power required for this system.

If we assume that the logic arrays are made up of 1000 x 1000 arrays of logic gates, a likely total size of the device would be 3 x 3 cm. This implies that the gates would be on 30 micron centers.

We define the time it takes to perform the primitive optical function as one block time. The simplest function which is performed on the whole array is a right angle turn by a mirror. We define this to be the block time: $t_b = L_b/c$, where L_b is the linear dimension of the array on nonlinear optical elements. With the assumption that devices would be 3 x 3 cm, we find the block time, $t_b = 100$ ps. From this we see that the simplest optical architecture would take four block times (Figure 9.1). In this system the output of the optical gate array is imaged on its input. The lens is assumed to be f/1. This system has a cycle time of 400 ps.

The optics required to perform a SS rule is composed of two parts: the recognizer optics and the scriber optics. The function of the recognizer is to determine if the bits required for a specific rule are present. The scriber uses the yes/no result from the recognizer to make a pattern to be substituted for the recognized pattern. We define the time (in block times) of the SS processes to be S. Examination of the operation of the scriber and recognizer optics reveals that both take approximately the same number of block times. So we assume the process of recognizing and scribing take the same amount of time (S/2).

The number of block times required to recognize a SS rule is dependent upon its complexity. Examination of the optics in [41] reveals that S/2 PB where B is the number of bits that makes up a symbol to be recognized and P is a constant related to the finite time required for imaging. This is because the input image must be split into B parts, and each split means that the optical path will be increased. We have assumed that the relationship between processing time and complexity is linear. From the paper, we will assume P4. We make this assumption because each split and recombine requires, as a minimum, two added beam splitters and some additional path length for imaging.

It is generally accepted that some relationship exists between the number and complexity of rules required and the time it takes to perform some operation. While it is easy to implement simple rules, more control bits are required. Further, more cycles are required. However, the execution of complex rules requires more time than simple rules since the optical path is longer. Again, since little is known about the quantitative relationship between rule complexity and the number of cycles required we will assume that it is linear. Furthermore, we will assume that, because of the longer cycle time required for complex rules, the exact implementation is relatively unimportant.

Since the processor will have many rules that may fire simultaneously, optics must be provided to split the image to go into the different SS optics. Figure 9.2 shows a modification of a concept of [41] which would perform this function. It uses a series of beam splitters to create multiple copies of the logic plane while maintaining a constant optical path through all the rules. Since each beam splitter takes a block time, the total number of block times to perform all the SS is:

$$T = S + 2R$$

where R is the number of rules.

To estimate the complexity of the problem, we begin by making some assumptions about the difficulty in implementing functions.

We have found that a set of about 25 different macroinstructions is required to execute the major functions of SPARO. We have assumed that an instruction will be represented by a symbol with 20 bits. Thus, to analyze the performance we can assume (quite arbitrarily, and perhaps optimistically) that about 25 different SS rules will be employed and that, on the average, 5 bits in each symbol would have to be recognized (B = 5). This means that

$$S = 2PB = 40$$

block times are required for one SS rule and that

$$T = 90$$

block times are required for one machine cycle. Thus, the total cycle time of SPARO then becomes

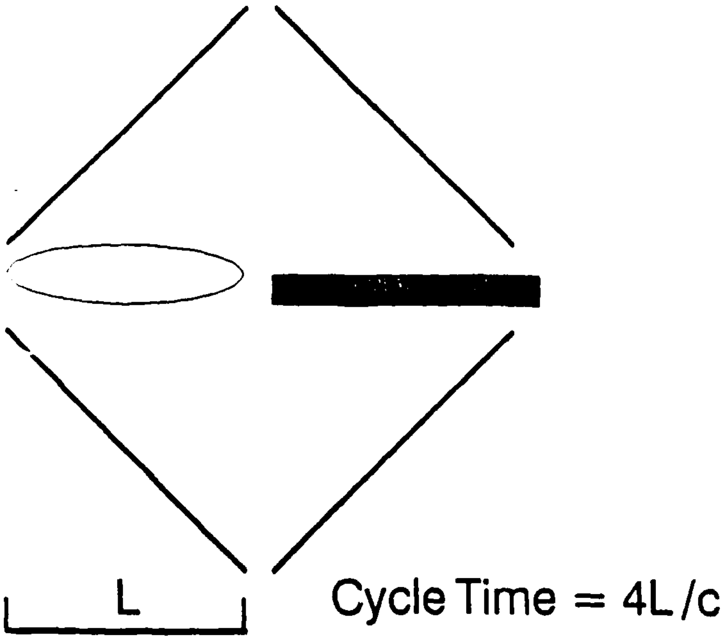
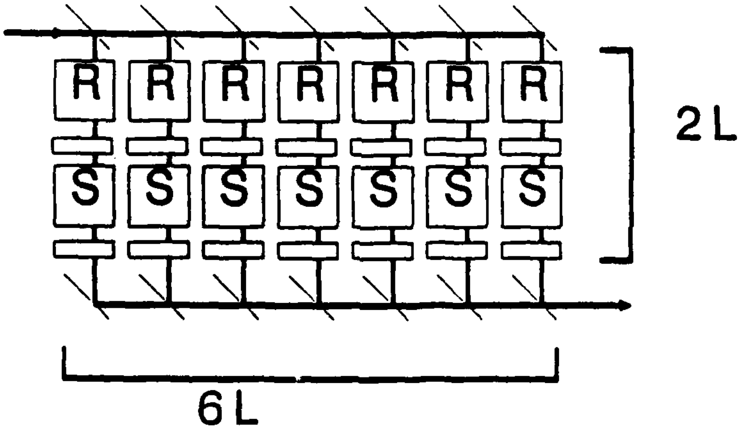


Figure 9.1 Calculation of cycle time



$$\text{Total transit time} = \frac{6L + 2L}{c}$$

Figure 9.2 Calculation of transit time

$$t = T\, T_b = 90 \times 100 \text{ ps} = 9 \text{ ns}$$

where we have assumed that the configuration of Figure 7.14 can be folded so that no additional time is required to connect the output to the input. This means that simple operations such as moves from register to register and movement of messages along the network take nearly 10 ns.

We have ignored the complexity of the gated interconnects in this analysis. This assumption is based on the contention that the movement of data required to recognize and set up a SS is comparable to the effort required to set up a transfer from one register to another. While this may or may not be the case, the interaction between the SS rules that control the gated interconnects will certainly increase the complexity of the overall optical system and consequently increase the cycle time. It would not be unreasonable to expect that a 10 to 20% overhead would result because of the need to increase path lengths to accommodate the connection of the SS and interconnect systems. Thus we can assume a cycle time for SPARO greater than 10 ns.

We do not believe that such an implementation is necessarily practical or optimal, but rather that it gives an order of magnitude estimate of the time required to perform the required functions.

This cycle time also implies that the processor can employ optical logic elements that switch quickly but take much longer to recover, a property present in many proposed optical logic elements [40]. This mode of operation is likely to be preferred in optical devices because it allows the use of materials that have larger nonlinearities, but relatively long recovery times.

Ideally, the switch would take place quite rapidly (on the order of picoseconds) so that no additional propagation time would be added to the system. However, allowing switches that operate in times comparable to the block time would only reduce the performance by a factor of two.

Another possible approach is for the system to be parallel in time as well as space. By using the processors in the PA in a pipeline mode, different graph reductions could be occurring at once. Thus, the implementation of each processor would be such that it would keep on executing while having pipelined its message requests for delivery by the network. Pipelining could also be implemented in the execution of the functional units as is commonly done in commercial floating point processors available today.

If we assume that logic gates with a total switching and recovery time of 10 ps are possible, and that the power dissipated could be removed, then the effective available performance goes up by a factor of 10^3. In SPARO this could instead translate to an increase in the number of possible nodes in the graph.

Unlike some parallel systems which cannot take advantage of the added processing power of a pipelined system, SPARO can use this extra power to have more nodes available in the graph (i.e., as memory) with the potential for more parallelism. This means that instead of being able to process graphs with only 10^3 nodes, SPARO would be capable of working on realistic systems with 10^6 nodes. With a system such as this, the processor would not require outside program or data memory and would be a complete computer.

Several problems present themselves in using this much parallelism. Foremost are how do each of the time slices physically communicate, and how do such a large number of processing nodes communicate effectively.

A simple time delay would be all that is needed for SPARO to send a message from one time slice to the next. It might even be possible (given the close time packing) to arrange for messages to be passed to the slice ahead by using a shorter route. All that would be required is to specify one location as the gateway to the other time slices and to include an optical delay (glass) equal to the time delay in its path (Figure 7.11a). Alternatively, the array could be delayed except for one location (Figure 7.11b). These elements could be employed to make the array look like one long (10^6 nodes!) network. More realistically, some other method of network routing would have to be employed.

Since in SPARO the extra processing power is being used as memory, it would be advantageous for the power consumption to be minimized. This could be accomplished by designing the system so that nodes which are idle (doing no processing) would not require many switching operations. In this way, if the amount of parallelism which can be exploited is not extreme, the increase in operating power is not significantly more than the un-pipelined mode, assuming switches with equal switch energies.

9.3 Parallel Reduction Rate

By using the results of our the SRR and the cycle time analysis, we find that for a 1K SPARO network, and a 10 ns cycle time, the reduction rate is only about 50K rps. There is almost three orders of performance loss due to the slow speed of the register-based bidirectional ring. This is comparable to the performance obtained in SKIM, the electronic graph reduction machine [42] that was rated at 100K rps. The SRR of SPARO is slower than the performance of NORMA [43] that boasts a performance of 250K rps (300K with more recent technology). The key factor that would set SPARO apart from electronic special-purpose machines is in the parallelism of the execution model. We therefore embarked on an evaluation of the throughput of the synchronous ring network that was originally proposed for SPARO.

Both analytical modeling and simulation show that the throughput of the ring network is severely limited. In the case of random message distribution, the throughput of the unidirectional ring and the bidirectional ring are bound asymptotically to 2 and 8, respectively. If messages exhibit locality, the performance improves to about 27 for a network of size 1K. Clearly, the simple ring network is a bottleneck in terms of serial and parallel throughput. This is because of the rather limited connectivity of the network. Only in cases where the processing is very local, as in image processing, can the performance of such a network be high. The conclusion is that while the ring network appears relatively easy in terms of implementation and complexity, it is poor in providing a reasonable performance.

9.3.1 Alternative Network Performance

Since the network appears to be the bottleneck, we have analyzed the performance of alternate network architectures. A network that is more sophisticated than the ring network but simpler than a fully-connected network is the replicated single-stage shuffle-exchange network. The shuffle-exchange network (SEN) appears attractive in terms of optics because of the relative ease with which the perfect shuffle can be implemented [44]. Purely from an architectural consideration, the throughput of a replicated SEN can be two orders higher than that of the ring network. Simulations of a fully-replicated SEN yield throughputs of about 385 for 1K network. Simulations of the SPARO architecture using a replicated shuffle-exchange network showed that the parallelism in a real CG for a parallel recursive factorial example yielded a reduction rate of 7M rps if a cycle time of 10 ns is assumed. The maximum parallelism in the application was only 10. Clearly, higher performance could be achieved if the application demonstrated a higher degree of parallelism.

The performance analysis reveals, not surprisingly, that the fine-grained parallel processing of SPARO introduces a severe bottleneck in the communications. Because of the lack of addressable optical memory and relatively limited power of optical processing (in the traditional electronic sense), we had to distribute the data and computation to design an unique and novel architecture. However, the parallelism in fine-grained computing called for very high message throughputs between the nodes of the PA. Simple network architectures such as the ring architectures proposed are unable to provide the desired throughput. While more sophisticated networks such as SENs can provide the high throughput, they are much more difficult to implement purely in optics.

9.4. SPARO Conclusions

Our approach to designing a symbolic optical computer based on the functional requirements of symbolic processing and the capabilities of optics have led to a unique architecture design. The architecture, SPARO, exploits parallelism at the lowest level of the combinator graph reduction computational model, resulting in a very fine-grained processor design. More importantly, to make the design feasible in optics, the architecture minimizes the stringent memory requirements that symbolic processing imposes. We have assumed that very fast cycle times of optical elements would allow the total computational throughput to be high because of the parallelism.

The general conclusions on evaluating SPARO are mixed in the sense that while an optical implementation appears possible, the feasibility and performance of such a system is questionable. The reasons for our conclusions can be summarized as follows.

First, current state of the art optics lacks scalable addressable memory devices that can compete with electronics. Addressable memory is central to symbolic processing, based on traditional or existing computational models, and therefore must be either provided or emulated in an optical symbolic processor.

Second, circumventing large addressable memories as done in our design of SPARO shifts the burden of computing to large switching systems. Based on our assumptions of implementing SPARO, the performance of the architecture is not clearly superior to an electronic implementation. To provide high throughput in a massively parallel architecture using optics, we require optical switching systems of complexities beyond the current state of the art.

Third, a fine-grained architecture resulting from a primitive memory technology demands high connectivity and high throughput of data or messages between processors. Current technology cannot support the large purely optical networks to provide the high throughput possible with massively parallel architectures. Recent work reveals that interconnection networks are also a serious bottleneck for massively parallel electronic machines such as the Connection Machine [45]. On a more optimistic note, the assessment of hybrid optical networks reveals that optics could potentially provide the high connectivity required of high-bandwidth high-density interconnection networks in massively parallel architectures.

The results of this work thus provide two broad directions in optical computing research. The long-term direction is in the development of conventional symbolic and numeric processors in optics. We find that the realization of such processors requires the vigorous development of integratable optical switches and memories. The near-term direction is in the area of massive connectivity for board-level computers where electronics is severely limited.

10. Analysis of Interconnection Networks for Sparo

10.1 Introduction

After the SPARO (Symbolic Processing Architecture in Optics) design had been completed, we embarked on a rigourous performance evaluation of both the serial and the parallel throughput possible. Our analyses revealed that while SPARO was novel in mapping the combinator graph reduction model onto a two-dimensional optical plane (where the evaluation could be done optically), the performance of the architecture was poor due to the inordinate sizes of the present-day devices required for solving non-trivial problems A more significant discovery was that the relatively simple shift register based ring network employed in SPARO was a severe bottleneck in achieving high throughput. Since fine-grained processing, as employed in the architecture of SPARO, depends critically on the performance of the interconnection network, this discovery motivated us to examine optical interconnection more closely. The emphasis of the program thus shifted from optical processing to optical interconnection networks.

The search for the ideal optical interconnection networks has led to the examination of different known networks such as hypercubes, shuffle-exchanges and the crossbar. Rigorous studies of the requirements of large parallel processing systems reveal that parallelism of connectivity is the key problem in implementing interconnection networks both in electronics and optics. Based on our analysis of the computing requirements and feasibility in optics, the single stage shuffle-exchange network (SEN) appears to be the interconnection network of choice.

Our discussion of interconnection networks is organized as follows. The next section presents the detailed analysis of interconnection networks for SPARO. Following this we examine the key issues in implementing purely optical SENs. The design of hybrid SENs, those implemented in a mix of optics and electronics, is then examined. For comparison, an electronic SEN design is discussed. The interconnection requirements of highly parallel systems are then summarized, and specific optical techniques evaluated with the goal of meeting those requirements.

Our initial analysis focuses on the performance of the ring network proposed earlier. The performance metrics, both throughput and waiting time of messages are derived analytically, and compared with simulated results. The other candidate interconnection network, the shuffle exchange network, which has been found suitable for optical implementation, was also analyzed. The results presented for the candidate networks are mostly from simulations. We show how shuffle-exchange networks, especially replicated shuffle exchange networks, can provide significant improvement in the message throughput and thus guarantee a greatly improved performance for SPARO.

The performance of the proposed optical architecture, SPARO (Symbolic Processing Architecture in Optics), was shown to be dominated by the performance of the interconnection network. In order to determine the expected throughput of the messages in the SPARO network, and thus the rate of reductions, it is necessary to analyze the network used. A shift register based ring network was proposed in the original architecture. It was expected that the systolic nature of the ring network would accomplish a high throughput of messages, and therefore provide fast execution of combinator graphs by using a high degree of parallelism. Our analytical model reveals, however, that even a bidirectional ring network of large sizes cannot provide significant parallelism. Thus, while the simple ring network is amenable for optical implementation, its performance is not acceptable. This motivated us to examine alternative candidates for the network. Among the networks found suitable for packet switching, the single-stage SEN and the binary hypercube are promising candidates. We have therefore examined and analyzed the SEN, and compared its performance against that of the hypercube and also the popular multistage interconnection network (MIN), the delta network. The reason for using the delta network is that it is considered a standard high-performance interconnection topology. Unfortunately, the implementation of a MIN is much more complex than the SEN. Our intent in the comparisons was to show that despite the simpler implementation of the SEN in optics, (or in optoelectronics) the performance of the SEN is quite competitive with a MIN.

We have provided the detailed analytical modeling for the bidirectional ring network. We derive the expressions for both the throughput and the waiting time. Two cases are considered in our analysis of the ring network performance: i) the case where messages for any processor node are equally probable, i.e., no locality is assumed, and ii) locality of messages are assumed whereby the probability of the message to a destination node varies inversely as the distance from the source.

Our analysis is based on the work presented earlier by Lawrie and Padua [46]. It is assumed that the conflicts in the SEN are resolved by giving priority to the message closest to its destination. We show, from the results of our simulations, that for a packet switching network, even a modest message generation can throttle the network. This underscores the inappropriateness of the loading factor used by Lawrie and Padua to characterize the networks. To alleviate the problem of increased waiting times and low throughput, we studied buffered SENs. Contrary to naive expectations, the introduction of buffers does not improve performance. We present arguments as to why such behavior is observed, since analytical modeling of buffered SENs with priority strategy for resolving conflicts is extremely difficult.

We also present our simulation results on replicated SENs. We show how replication of SENs can dramatically improve throughput. Based on our results, we show what order replication would be recommended, given performance and cost constraints. The other alternative to replication is to use an enlarged network or a network that is about four times as large as the number of processors to be connected. The choice of replication or enlarging the network would be determined by the relative difficulty of merging multiple networks (in the case of replication) and the maximum size of the network that can be implemented (in the case of enlarging the network).

Finally, we present a summary on the performance analyses of hypercubes and delta networks for comparison with the SENs. The comparison is based on our simulated network results as well as the theoretical work done by other researchers.

10.2 Performance of ring networks

Before presenting the model used to represent the ring network, we state the assumptions made in our analysis. We also preface our assumptions with a brief description of the network.

10.2.1 Principle of operation

The register-based network originally designed for SPARO, purely by serendipity, looks quite similar to that proposed earlier for the ZMOB parallel processor [39] intended for image processing applications. The SPARO network is composed of at least 1024 registers (the size being determined by the size of problems for which the architecture is targeted) connected in a conveyor belt fashion. Each stage or register is associated at any time with a single processor node. There are thus 1024 processor nodes. Each processor node can receive or send a message by accepting a message from or loading into its associated register in the network. Messages are delivered by the network by shifting the registers in a conveyor belt fashion. Since each message has a destination address, the message reaches its destination when the processor node address matches that of the message. Unlike the ZMOB network, the SPARO network is bidirectional. The network is assumed to recognize the direction which results in a shorter delivery path for a message. The analysis of unidirectional and bidirectional network is only trivially different.

In terms of operation, the following three-phase sequence is followed in SPARO:

i) Each processor in the network examines if its message output buffer is empty, that is, if the previous message has been delivered. If the output buffer is full, the processor cannot load its new message into the buffer and therefore enters a wait state. Otherwise, the processor will load the output buffer with its message and continue processing.

ii) The ring network shifts and the ring register associated with each processor examines if it has a message to deliver to the processor. This is done by comparing the destination address of the message to that of the processor. If the addresses match, the message is delivered to the input buffer of the processor.

iii) The processor checks if the associated ring register is full, that is, the message in the register is meant for another processor. If the register is empty, then the message in the output buffer is loaded into the ring register. Otherwise, the processor waits to offload its message in the next network cycle.

Since in phase (iii) the processor buffer cannot be emptied, the processor cannot generate more messages. This allows the ring to proceed uninterrupted at its full speed, and also ensures that no messages are lost. This assumption also implies that the message generation rate is influenced by the loading of the network. (The assumption reflects, especially in the case of fine-grained processing, that a processor operates on simple sequential tasks and cannot proceed until the previously sent message has been delivered and a response message has been received.) We now examine the analytical model of the *ring network* in brief.

10.2.2 Analytical model of the ring network

Figure 10.1 shows the representation of an individual stage of the network and its communication with the processors. We define below the following parameters that are used to define and compute the throughput of the network.

$N = 2n$: total number of nodes in the network

$p(i)$: probability that the destination of a message is i shifts away from source

mg0: rate of message generation at each node under no loading restrictions

mg: effective rate of message generation at each node

m: total rate at which messages arrive at a node via the network

Pt: probability of termination of a message arriving at a node

In the first part of our analysis, we consider the case that messages in the network are equally likely to be delivered to any node in the network. This contrasts with the case that the messages exhibit locality, or that the probability of accessing remote destinations is lower than those nearer to the source node. In the equiprobable case, the probability of generating a message with a destination that is i shifts away is independent of i. This probability is $2/N$ for a bidirectional ring ($1/N$ for a unidirectional ring) where $N = 2n$ is the ring size.

We can calculate the rate m at which messages are traversing the network. If mg be the *effective* probability of message generation in each processor, and Pt the probability that a message has terminated or reached its destination, then in the steady state, the rate of generation of messages, mg, will be equal to the rate of consumption $m\ Pt$. Then, in a bidirectional ring,

$$m = mg\ /\ 2\ Pt$$

In the case of the unidirectional ring, m is twice that of the bidirectional ring.

Note that the above expression involves the effective message generation rate mg and not the actual message generation rate which we denote by $mg0$. This modification reflects the fact that the effective message generation rate depends on the load on the network. As traffic on the network, indicated by m, increases, the message generation rate will decrease. Thus, $mg < mg0$. The effective message generation rate can be computed by knowing when the buffer in the processor is full. If $q1$ and $q0$ be the probability that the buffer is full and empty, respectively, then we can find $q1$ using the relation

$$q1 = a01\ q0 + a11\ q1.$$

where $a01$ and $a11$ are the probabilities of transition from $q0$ to $q1$ and vice versa. $a01$ is thus the message generation rate when the network register is occupied, or

$$a01 = mg0\ m(1 - Pt\ ).$$

$a11$ is the probability that a non-terminating message arrives at the processor node or

$$a11 = m\ (1 - Pt\ ).$$

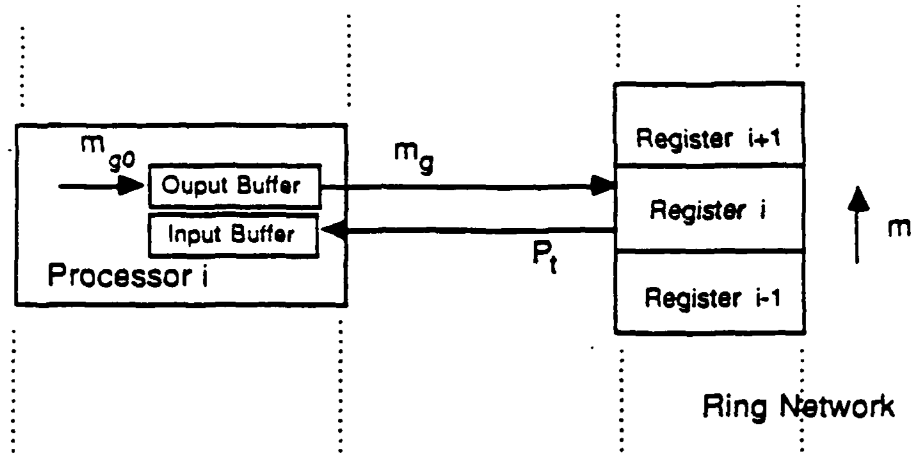


Figure 10.1 Processor/ Ring network interface

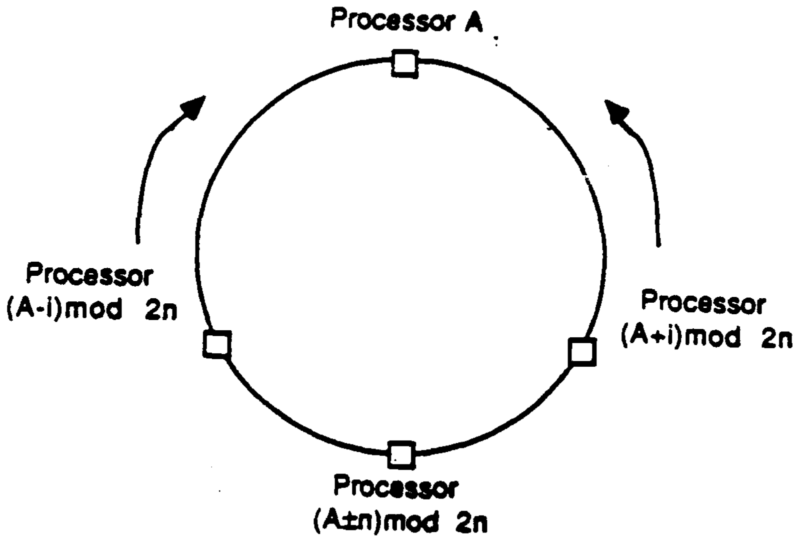


Figure 10.2 Message sources for bidirectional rings

Since q0 = 1 - q1, we can show q0 and q1 to be

$$q0 = [1 - m(1 - Pt)]/ [1 - m(1 - Pt)(1 - mg0)]$$

$$q1 = m\ mg0\ (1 - Pt)/[1 - m(1 - Pt)(1 - mg0)]$$

Using the expression for m previously derived, we find that m is the solution to the following equation.

$$m^2 [2\ Pt\ (1 - Pt)(1 - mg0)] - m[2\ Pt - mg0\ (1 - Pt)] + mg0 = 0$$

In the case of the unidirectional ring, the corresponding equation for m can be obtained by removing all occurrences of 2 from the above equation.

It can be shown that one of the roots of the above quadratic equation for m is not viable. The legitimate value of m is found to be

$$m = a - (b/c)^{1/2}$$

where a, b and c are defined to be:

$$a = 2Pt + mg0(1 - Pt)$$

$$b = [2Pt - mg0(1 - Pt)]^2 + 8Pt(1 - Pt)mg0^2$$

$$c = 4Pt(1 - Pt)(1 - mg0)$$

To calculate the throughput, we need to determine Pt, the probability of termination of any messaje. Pt in turn can be computed if the distribution of messages in steady state is known. By steady state distribution we mean the distribution probability of messages with different destination distances, from 1 to n (N for the unidirectional case).

To find the termination probability Pt, we first derive the distribution probability for messages with random destinations. The method used for deriving the message probabilities is similar to the one used by Abraham and Padmanabhan [47]. Note from Figure 10.2 that there are two possible message sources (processor nodes) in a bidirectional ring that are i shifts away when i < n, but only one when $i = n$. Suppose two shifts have taken place in the network. The distribution of messages can be derived as follows. The number of messages that require $(i - 1)$, i _ n, shifts to reach its destination is 2 + 2. (The first term corresponds to the number of messages generated in the second shift that require $i -1$ shifts, while the second term corresponds to those that need the same number of shifts but were generated in the first shift.) Note that the 2 appearing in each term, except for the destination n away from the source, is due to the fact that we are considering bidirectional networks. Since the maximum number of shifts required for the ring network is n, the distribution of messages with different required shifts reaches steady state after n shifts. The number of messages requiring $i - 1$ shifts is $2 (n - i) + 1$.

To derive the normalized probability, we need to find A where

$$A = S [2(n - i) + 1] = n^{2} \text{ where i varies from 1 to n}$$

Thus, the probability that a message requires $i - 1$ shifts to reach its destination:

$$p(i -1) = \{2(n - i) + 1\} / n^{2}$$

The termination probability Pt is the probability that the destination of the message is 0 shifts away. This can be found by simply setting $i = 1$ in the expression for $p(i - 1)$, which gives $Pt = (2n - 1) / n^{2}$. The termination probability for the unidirectional case can be shown to be $1/n$.

When messages exhibit locality, the same analysis can be carried out, except that the probability of messages requiring i shifts decreases as i increases. In case of a harmonic distribution of messages, the probability of a message requiring i shifts is inversely proportional to i .

The procedure to derive the expression for $p(i - 1)$ is exactly similar to that used in the earlier case, except that messages requiring different number of shifts are assigned different probabilities. Thus, $p(i - 1)$ can be written as:

$$p(i - 1) = S (2/j - 1/n) / A \text{ where j varies from i to n.}$$

where A can be shown to be:

$$A = S\ S (1/j - 1) \text{ where j varies from i to n while i varies from 1 to n}$$

Setting i = 1 in p(i - 1), we obtain Pt

Pt = S (2/j - 1/n) / A where j varies from 1 to n

Neither the numerator and denominator can be expressed in closed form.

10.2.3 Throughput of ring networks

The throughput is defined as the average number of messages delivered at the end of each cycle or shift of the ring. Since m is the rate at which messages arrive at a node via the network, the number of messages delivered at a node is m Pt. Since there are N nodes in the network, the total throughput, denoted by T, can be expressed as follows.

For a bidirectional ring T = 2 m Pt N = 8 m (N - 1)/N
For a unidirectional ring T = m N Pt = 2 m

In the case of locality, the throughput expression for the bidirectional ring cannot be expressed in closed form. It can be seen that the throughput for no locality asymptotically levels off to 2 and 8 for the unidirectional and bidirectional rings, respectively. Thus, the throughput of unidirectional rings can be quadrupled at only twice the cost.

Figures 10.3 and 10.4 graphically depict the throughput for bidirectional rings when messages have random and local destinations. Figure 10.3 shows the near-exponential increase in the total delay time (theoretical derivations are not included here) which is composed of the waiting time in the buffer and the transit time over the network. As can be seen, the analytical results agree closely with the simulated results. It is of interest to note that when messages exhibit locality, the throughput reaches as much as 27. This is more than three times the throughput of rings with no local messages.

10.2.4 Conclusions

We have presented an analytical model to evaluate the throughput of ring interconnection networks in message-passing environments. Although the model is relatively simple, it effectively shows the serious limitations of the register-based ring networks. Clearly, from Figures 10.3. and 10.4, for processors using message passing for communications, the ring network cannot provide an acceptable throughput for more than about 16 processors.

We next evaluate other network topologies as possible candidates interconnecting the processors in SPARO.

10.3 Performance of single-stage and replicated shuffle-exchange networks

The first alternative topology that we examined in detail is the shuffle-exchange network. We also examined the potential of employing replicated shuffle-exchange networks which have been used in electronic network designs to improve the performance of the single-stage network. Although analytical models for predicting the performance of these networks have been studied, the question of what the desired level of replication should be, has been left unanswered. We therefore examine, from an architectural perspective, which of the following possible networks is desirable:

i) a single shuffle-exchange network (SEN),

ii) a full multistage interconnection network consisting of $\log_2 N$ stages when N processor nodes are to be interconnected, or

iii) replicated SENs where the degree of replication is between 1 and $\log_2 N$.

10.3.1 The shuffle-exchange network

We will initially consider the single stage SEN. The operation of the SEN that we are considering is described in detail by Lang in [4,8]. A one-stage SEN contains $N = 2^n$ registers or buffers, indexed from 0 to N - 1, when N processors are connected to the network. Figure 10.5 shows the SEN for connecting 8 processors or network modules. To deliver messages from the processors, the network operates by cycles. In each cycle, a message and its destination tag (binary address of the destination processor) passes through an exchange element and then undergoes a shuffle permutation. If the destination processor is reached, the message is delivered to the processor, or else it is injected back into the network. It takes at most n periods for a message to reach its destination.

The shuffle stage of the SEN is used to realize a perfect shuffle among the N elements. The perfect shuffle can be described by the following relation:

$$S(i) = (2i + \lfloor 2i/N \rfloor) \bmod N$$

where S(i) denotes the destination of the message from the ith processor due the shuffle permutation and ¦ ¦ represent the lower

Figure 10.3 Ring network throughput for random message distribution

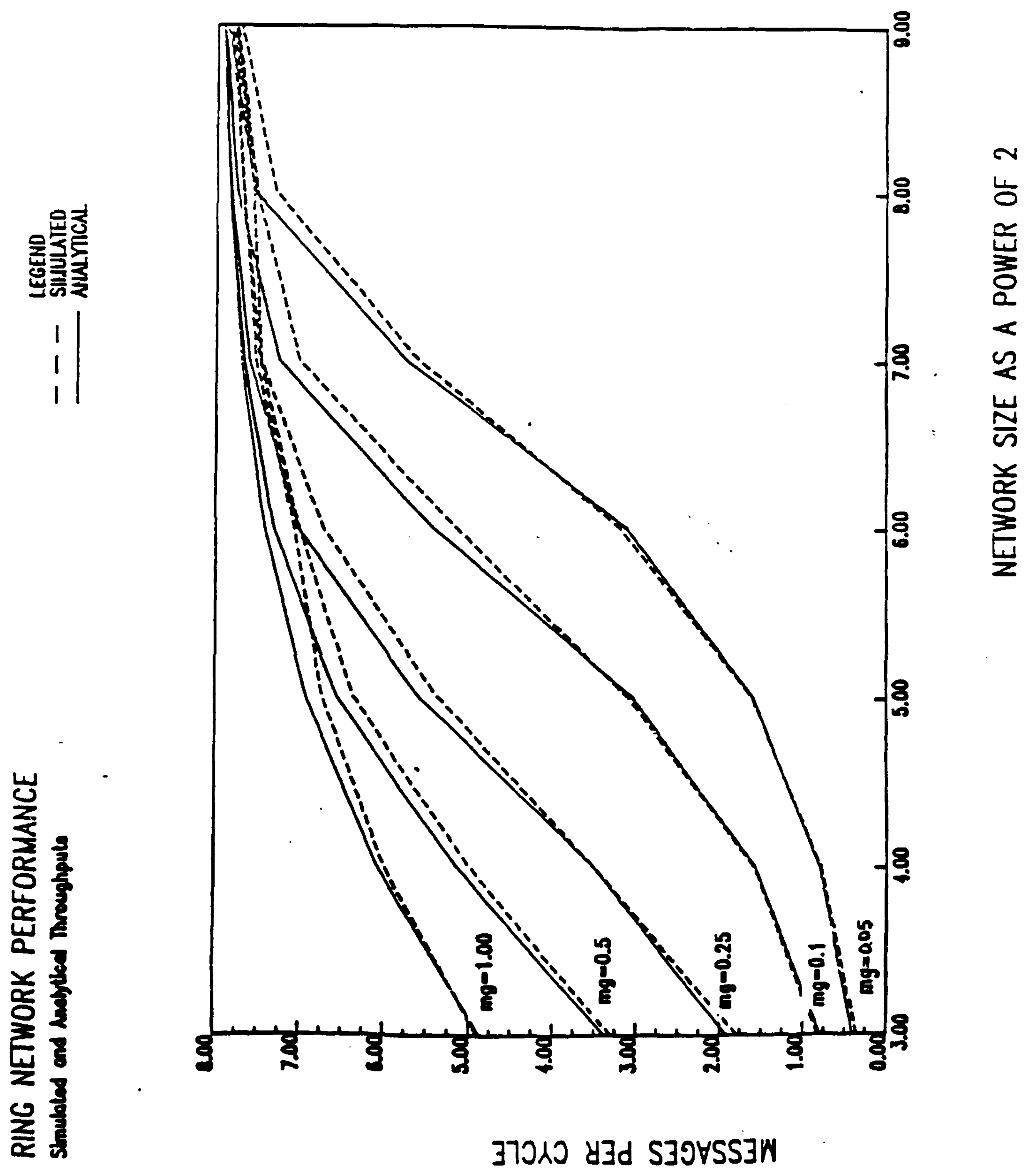
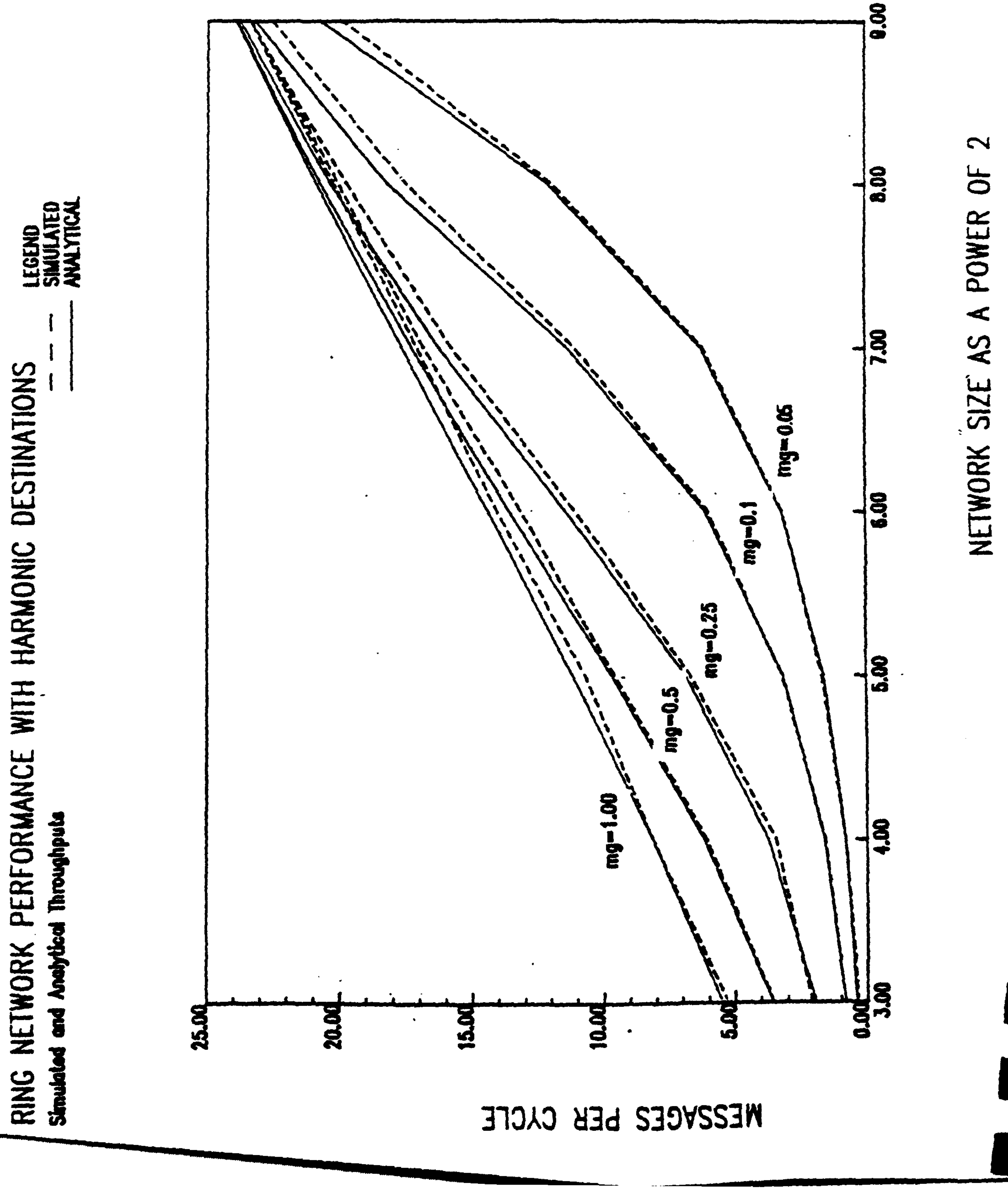


Figure 10.4 Ring network throughput for harmonic message distribution



integer ceiling of the enclosed expression. Figure 10.5 can be used to verify the shuffle stage for N = 8. The second part of the SEN consists of N/2 exchange elements. The purpose of the exchange element is to exchange the destinations of messages arriving from two adjacent processors. The exchange is done based on the value of a control bit. Depending on the control bit, the exchange element will either exchange the position of the input messages or leave them unchanged. Instead of using separate control bits for routing a message through the SEN, one can conveniently use the destination tag method [49] for setting the control of the exchange elements. In this method, if b_n b_{n-1} ...b_1 be the binary representation of the destination processor, then during the (i - 1)th period, bit b_i is used to set the exchange element. Depending on the controlling bit, the switch will either exchange, i.e., cross-connect the input and the output message packets or let them pass through undisturbed. Clearly, since each input message can provide the switch setting independently, there is a fifty-fifty chance of conflict when two messages arrive at an exchange element.

10.3.2 Operational model of the SEN and its analysis

A good operational model and its analysis has been presented in [46]. Here we will give a brief description of the model to motivate the study of replicated SENs.

In the normal operational mode, several messages will be circulating in the network. Both from an analysis as well as from an implementation viewpoint, it is easier to consider the synchronous operation of the network. Synchronous operation of the network implies that the exchange elements and the registers (that hold the messages to be injected to the network) are latched simultaneously. On the average, messages in a SEN can be delivered within 2n cycles. There is no upper bound on the number of cycles required since in each pass through the network a message may get blocked, due to possible conflicts arising at the exchange elements of the network. The exchange element resolves this conflict by allowing one message to go through to the proper destination. If messages cannot be dropped, then the message that loses in the conflict resolution is routed via a longer path, thus increasing the 'delay time' or the number of cycles required to deliver it. Two schemes for resolving conflicts are often used:

i) random selection, and

ii) closest first selection.

In the random selection scheme, as the name implies, the message chosen for routing to the proper destination is chosen randomly. The message that loses in the random selection starts afresh in the routing cycle. To represent the status of a message in the routing, a counter is associated with the message. The counter is initially set to one. If the message is successfully

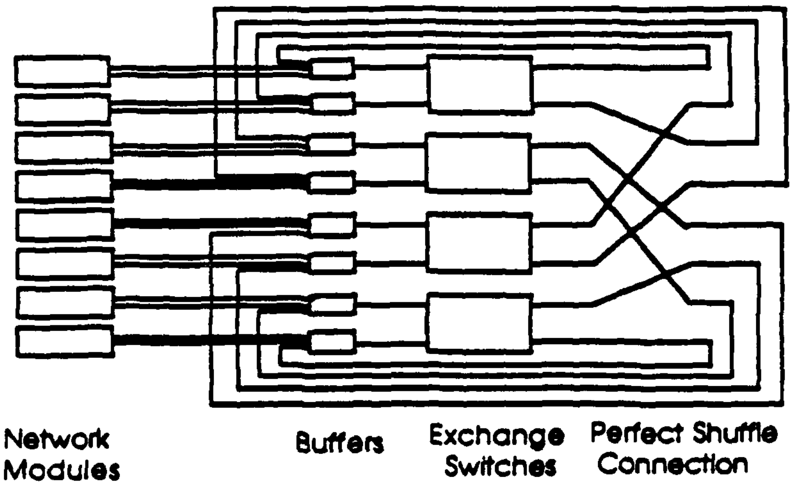


Figure 10.5 Single stage SEN for 8 PEs

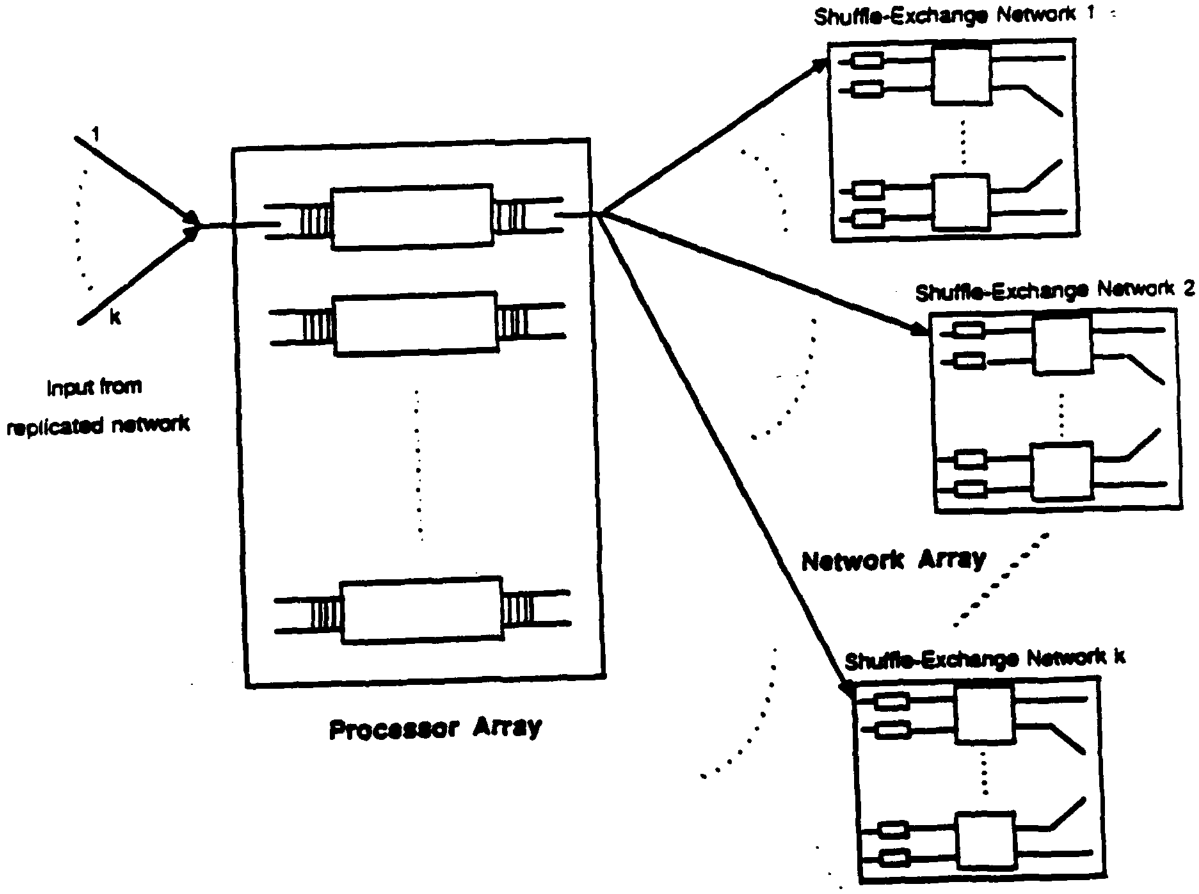

1
k
Input from replicated network
Processor Array
Shuffle-Exchange Network 1
Shuffle-Exchange Network 2
Network Array
Shuffle-Exchange Network k

Figure 10.6 Replicated SEN

routed in one pass or period of the SEN, its counter is incremented. However, if the message loses during a conflict resolution, its counter is reset. Thus, a message reaches its destination when the counter value is n+1. In the prioritized case of the closest first selection, the conflict is resolved by selecting the message with the larger counter value (randomly, if there is tie). Results from [46] show that, as expected, the message delay is smaller and the throughput is higher (for small loads) in the second case. Therefore, we have focussed our attention on SENs utilizing the prioritized conflict resolution scheme.

Previous analysis of the SEN considered the state transition of the counter to determine the probability that a message has been delivered (i.e., the probability that the message has a counter value of n+1). This would then yield both the throughput and the expected number of periods (delay) that a message stays in the network. Unfortunately, the authors could not find a closed form expressions for these metrics for the prioritized selection case. Instead, numerical solutions have been provided for different 'loads', where the load is defined as the fraction of active messages to the total number (N) of processors. Note that the load is distinct from the rate of message generation by the processors. The results showed that for loads below 0.25, the expected delay is $1.5\log_2 N$ periods. For larger loads, the expected delay rises while the throughput falls off, especially for large N.

Previous researchers [46] have mentioned briefly the use of replicated SENs for improved throughput when the load is high. A detailed study of the effectiveness of replicated networks has not been undertaken. We have chosen to examine the advantages of replicated SENs, in the operational mode described above, in greater detail.

10.3.4 Replicated Shuffle-Exchange Networks

Figure 10.6 illustrates a replicated SEN (RSEN). If a k-replicated network is used, then k networks are connected in parallel. A message to be delivered can be routed to any available network. Similarly, at most k routed messages can arrive at the input of a processor. For each of the k networks, the effective number of processors generating messages is at most N/k. Thus, the expected load in each SEN in the k-replicated network is at most 1/k. In our study of RSENs, we have limited k to be less than or equal to n since any message takes at most n passes to be routed if no conflict occurs.

Our focus in the study of replicated networks was on the performance of k-replicated SENs for different values of k and for different message generation rates. The advantage in using RSENs, besides the increased throughputs, is the flexibility of varying the amount of replication. As Figure 10.8 shows the level of replication can be increased from 1 to 10 for a 1024 network for increased throughput. The cost and the desired performance will decide what level of replication is most suitable.

We next present the results on the performance of single SEN with different network sizes, and the performance of k-replicated networks for different k.

10.3.5 Performance of single and replicated SENs

Two different sets of results have been obtained by simulation. First, we have examined the effect of the size of the network on the throughput for a fixed message generation rate. This has been done for a single SEN for the purposes of studying the effect of size on performance. Second, we have considered the effect of the degree of replication on the both the throughput and the delay time.

The performance of the single-stage SEN and the RSEN is given graphically in Figures 10.7 and 10.8. Figure 10.7 depicts the throughput and delay times for delivery of messages in a single-stage SEN of various sizes up to 1024. The message generation rate considered is only 0.25 since rates higher than this lead to a fully loaded network. A fully loaded network exhibits a very large number of conflicts resulting in very few deliveries per cycle. According to our simulations, a 1024 network has a throughput of only 40 when 256 processors can generate a message on average (message generation rate of 0.25). The RSEN performance (Figure 10.8) shows the throughput can be increased using replication. Note that the throughput shown in Figure 10.8 is the throughput per individual network. Thus, the total throughput for a 10-replicated network is greater than 380 when the message generation rate is 1.0. The dramatic increase in throughput in RSENs is due to the decreased loading in each network which results in fewer conflicts than in the single-stage SEN.

Another result obtained from simulations which influences the implementation is the effect of the delivery schedule of messages. In the normal delivery scheme, a message is delivered to its destination when the counter associated with the message reads $\log_2 N$. We had expected that delivering messages on the basis of the comparison of the destination address with the address of the node at the end of each cycle would be more efficient. However, simulations show there is no perceptible difference in the total delay time of messages or the throughput when the second delivery scheme is adopted. The lack of difference can be attributed to the effect of conflicts that erode any advantages expected in the scheme using address comparisons.

We have also examined the use of buffers in single stage SENs to improve performance. We examined the effect of first-in-first-out buffers to queue arriving messages being

Figure 10.7 Performance of single stage SENs

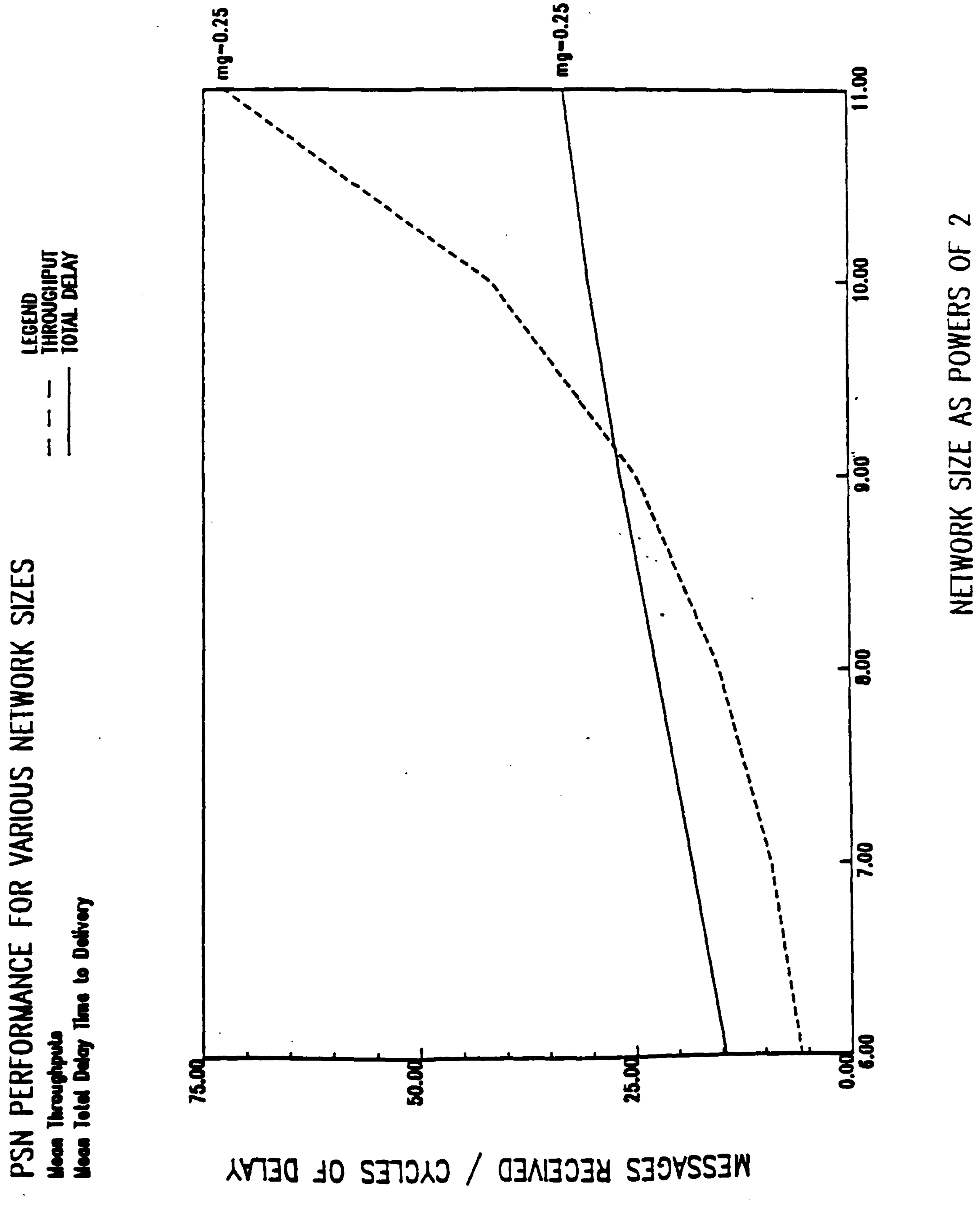
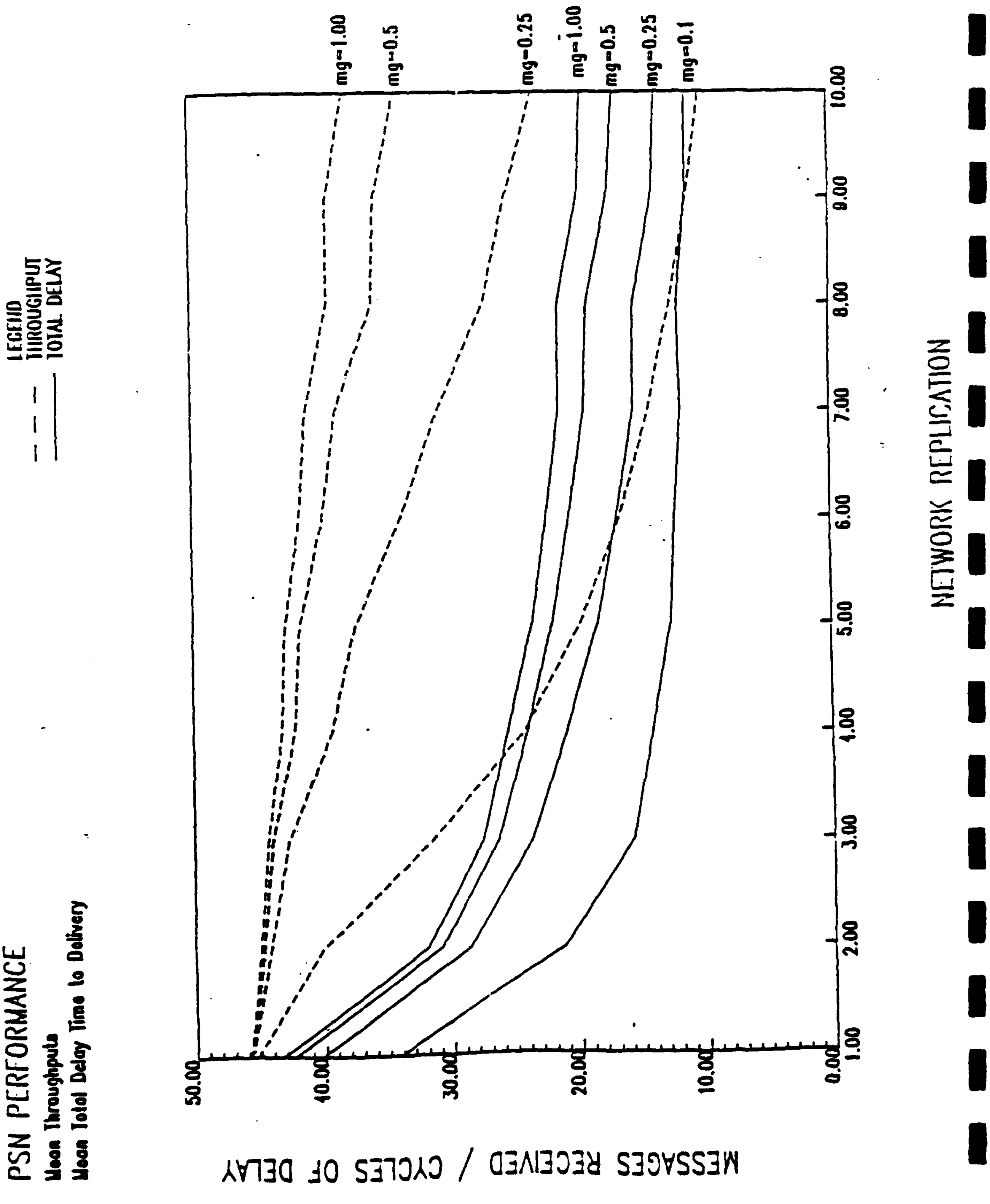


Figure 10.8 Performance of RSEN



injected in the network. Interestingly, the performance of buffered SENs, even for modest loads, degrades drastically as the number of buffers increases. This is because message delays increase almost exponentially while the throughput remains almost constant once the network is fully loaded. While the results may not be obvious at first, the results can be explained as follows. When buffers are used, processors can generate messages while buffers are not full even when the network routes messages through them. At some point, when all buffers are non-empty, the network essentially has a load of 1. From earlier results [46], we know that the performance degrades as the load on the network is increased above 0.25.

A further problem in buffered single stage SENs is the problem of routing messages into a processor whose buffer is full. One approach is to provide handshaking capability between source and destination processors. However, in a single-stage SEN, a chain of up to n handshakes maybe required (as in the multistage SEN). When a buffer is full and a message has to be accommodated in the input queue, some message in the buffer has to be removed to some other processor. Such a scheme will require more complex control and therefore a buffered SEN does appear attractive.

To examine the performance of SENs and RSENs more objectively, we examined other candidate networks under similar conditions of size and load.

10.4 Comparison of RSENs with other networks

The strength of RSENs can be judged best on the basis of their performance relative to other well-known networks. We have therefore examined a number of interconnection networks that are commonly used in electronic parallel processing architectures.

10.4.1 Comparisons with multistage interconnection networks

Multistage interconnection networks (MINs) are very popular in implementing large parallel processors. An example of such a network in a commercial machine is the BBN Butterfly. Figure 10.9 shows the topology for an 8 X 8 MIN. To compare the RSEN with the MIN, we have relied on the results on unbuffered delta networks (one class of MINs) given by Patel [51] and on the results on buffered delta networks by Dias and Jump [50]. Although both sets of figures are obtained from analytical models, Dias and Jump have verified their results by simulation.

The results for a MIN show that for a 1024 10-stage MIN, the normalized throughput (that is, the ratio of the absolute throughput to the total number of processors) is about 0.2 for an unbuffered network and 0.39 for a network with a single buffer. These figures translate to a throughput of 205 in case of the

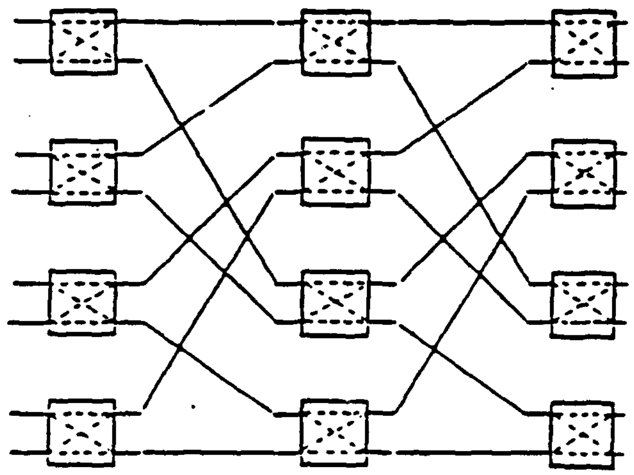


Figure 10.9 8x8 Multistage interconnection network

unbuffered network and a throughput of approximately 400 for a network with a single buffer. This is comparable to the throughput of a RSEN (380) composed of 10 networks. The normalized delay time to deliver a message in the MIN is about 1.5 or 15 cycles for a 10-stage network. This also compares well with the 15 - 20 cycle range observed in the RSEN.

In comparing the RSEN with the MIN, note that while the MIN has multiple (log$_2$N) switching stages, the processors in a k-replicated RSEN must be able to accept up to k messages on its input port. However, in the case of a RSEN there is an added flexibility of using less than log$_2$N shuffle-exchange networks if less than maximum throughput is acceptable.

10.4.2 Comparisons with the hypercube

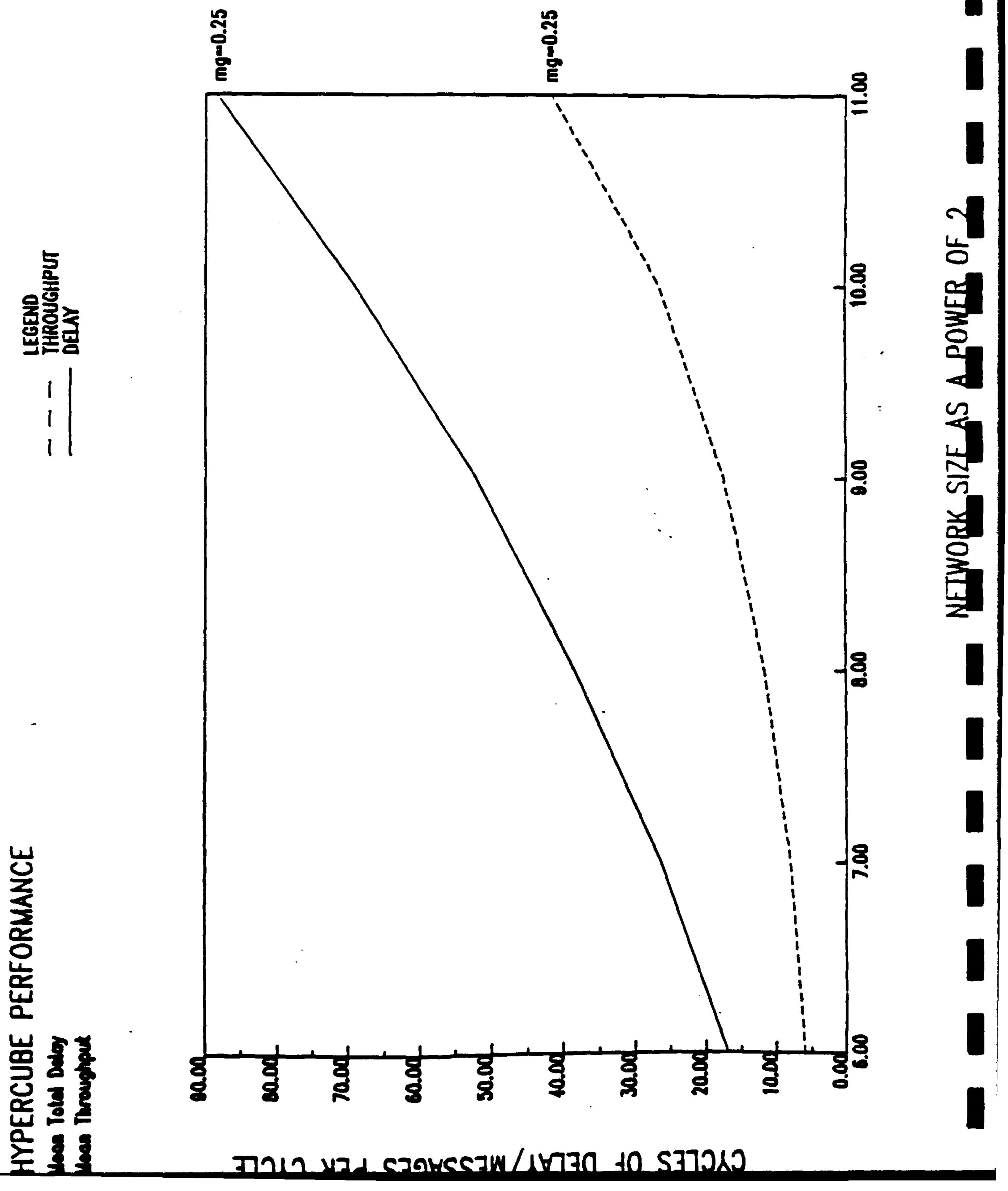
To compare the performance of RSENs with another popular interconnection network of comparable size, we have examined the hypercube topology. The hypercube has recently become popular by making its appearance in two commercial machines the Connection Machine (CM) [52] and the Intel Hypercube. Figure 10.10 shows the topology of a hypercube of 4 dimensions. In our analyses, we have considered a 10-dimensional hypercube.

As in the CM, each node in the hypercube is assumed to possess a routing buffer of length k (0 _ k _ log$_2$N = n). The routing in the hypercube is determined by forming the bit-by-bit EXOR of the source and destination addresses. The bit positions of the result indicate the dimensions along which routing takes place. The network operates synchronously with one dimension being active at a time. There is no set transfer sequence amongst the dimensions for a message to be routed.

When two processors are connected during some dimension cycle, each processor examines its own buffer to check for messages that need to be transferred. If none are found, a processor checks to determine if its buffer is full. If both processors have messages to transfer, a two-way transfer takes place. If only a single message packet needs to be transferred, the transfer is possible only if the buffer of the destination processor is not full. Two modes of operation are possible when considering the delivery of messages. In the first, both message generation and delivery are allowed in every dimension cycle. In the second, there is an upper bound on the number of messages that can be generated and delivered in every n dimension cycles or one network cycle. The second mode of operation is followed in the CM.

In the first mode of operation, since messages can be delivered in each dimension cycle, one expects a smaller waiting and therefore a smaller delay time. While this should result in better performance, the control is expected to be more complex and the dimension cycle would be longer than in the CM mode. The

Figure 10.11 Performance of the Hypercube as a function of size



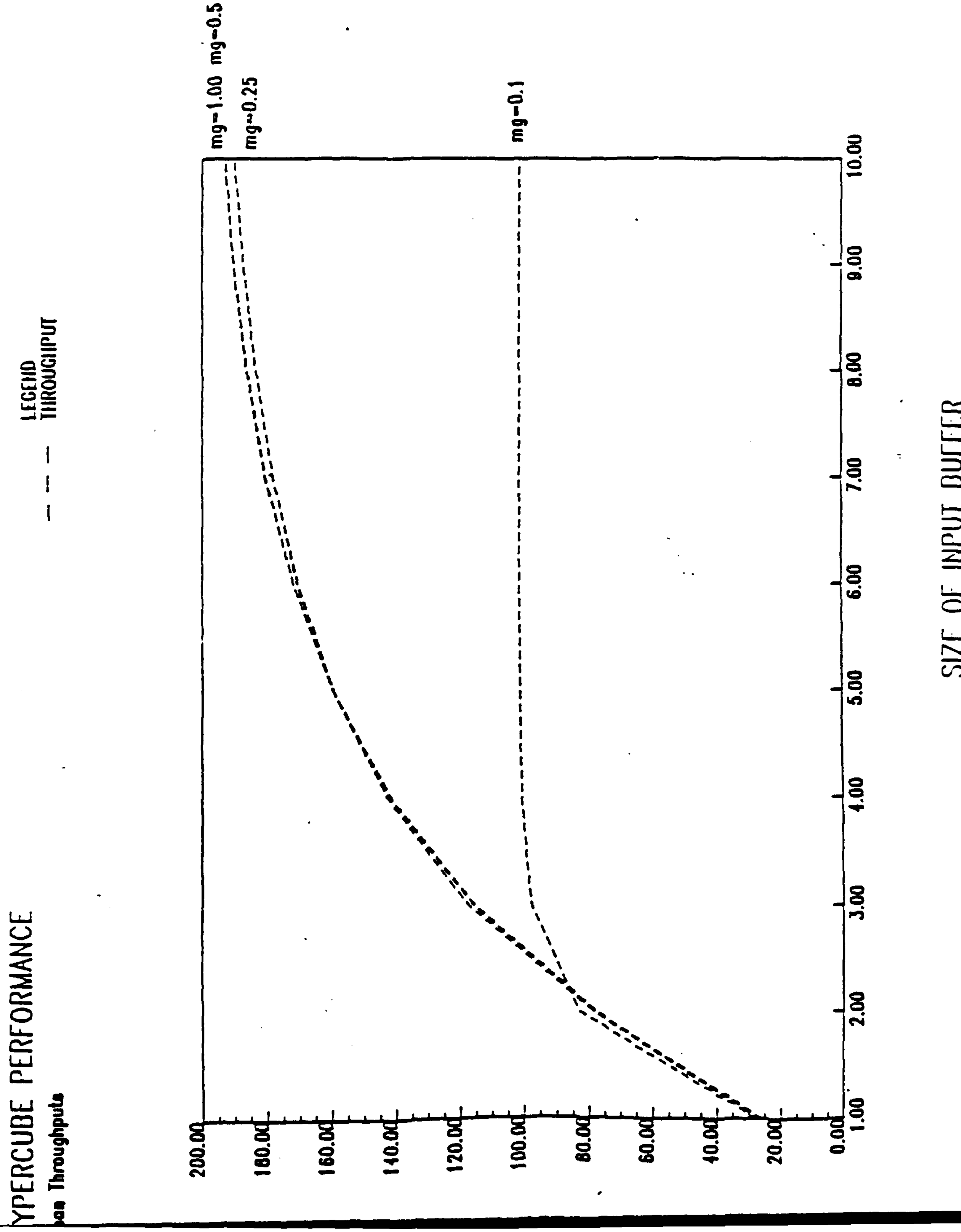
network cycles in the two modes of operation of the hypercube therefore have different meanings. In the first mode where deliveries are allowed in each dimension cycle, an individual dimension cycle is longer to allow for message deliveries. In the CM mode of operation, the network cycle is composed of simpler dimension cycles during which messages can only be transferred. All deliveries take place at the end of the network cycle. While we focussed our attention on the first method, we simulated both modes of operation for comparison.

Figure 10.11 shows how the throughput varies with the size of the hypercube when a single buffer is used at a message generation rate of 0.25 per cycle. We have assumed the CM mode of operation. With a message generation rate of 0.25, the throughput is only about 25 for a hypercube of 10 dimensions. The poor throughput results from the overflow of the input buffer in each dimension cycle when more than one message have to be transferred across pairs of processors. Figure 10.12 examines the throughput for a 1024 hypercube as a function of the buffer size. The maximum throughput for such a network is about 190 when a buffer of size 10 is used.

In the CM mode of operation, where 1 message generation and 1 message delivery was allowed (that is, we allow 1 message to be generated, if the buffer is not full, and 1 delivery, if any message was generated, at the end of each network cycle), the throughput reaches a maximum value of about 1000 per network cycle. In either mode of operation of the hypercube (a maximum of 200 per dimension cycle), we believe that the RSEN is very competitive (385 per single cycle). It must be noted that it is difficult to make an exact quantitative comparison between the two networks since the RSEN cycle of operation is different from the dimension cycle in the hypercube. As pointed out earlier, the dimension cycle in the CM mode of operation is shorter since no message deliveries occur until a network cycle is completed. Actual implementation issues must be considered before accurate comparisons can be made.

In terms of optical implementation, the SEN appears more attractive than the hypercube which must use large buffers for each node in the processor. On such simple first order analysis, the SEN cycle would be expected to be shorter than that of the hypercube. Thus, given that the two networks are competitive on the basis of throughputs, the RSEN would appear to be a better candidate.

Figure 10.12 Hypercube performance as a function of buffer size



11. Implementation Issues in Optical Shuffle-Exchange Networks

Our analysis of interconnection networks, based mostly on performance, reveals the SEN to be competitive with other commonly used networks with low loads (<0.25). For larger loads, replicated SENs or RSENs are very competitive with other commonly used topologies. The SEN also appears more attractive than other networks because of recent work on the optical implementation of the perfect shuffle. A number of different optical shuffle implementations have been proposed. Lohmann [45] and Midwinter [53] initially showed how the perfect shuffle can be implemented very effectively using passive optical elements such as lens and prism combinations or holograms. Eichmann and Li [54] have later shown an even more compact implementation in optics which reduces the total optical path length from Lohmann's approach by a factor of 6. Their results indicate that the channel spacing d and the spot size a are the limiting factors.

$$d = D / \{N - 1\}, \; a < D / \{2(N - 1)\}$$

where D is the aperture of the lens used.

With a 50 X 50 sq. mm aperture lens, the optical perfect shuffle can handle as many as 40,000 light channels. The channel spacing and spot size is assumed to be 0.25 and 0.1 mm, respectively. While using bulk optics, as proposed in [54] may lead to larger-sized SENs, alternative holographic approaches and guided approaches may be used to accomplish the same shuffle permutation. We will examine these alternative approaches in the next section.

To make the design process of the optical SEN more tractable, we examined the proposed optical designs and also extracted the key requirements for the critical portion of the SEN, the exchange switch.

11.1 Existing optical shuffle-exchange designs

There has been increasing interest recently in the implementation of the optical perfect shuffle for sorting networks but very few efforts on the exchange switch implementation. Here we outline the work that has been reported thus far.

11.1.1 Perfect shuffle

Lohmann [44] appears to have been the first to present an optical perfect shuffle design. He proposes a setup using prisms and lenses. The input elements are divided into two halves, upper and lower, which are stretched in one direction to match the size of the original inputs. The stretched halves are then recombined by interlacing to achieve the perfect shuffle of the inputs. The outputs on recombination appear in reverse shuffle order but can easily be 'unreversed' by standard optical means. The total optical path length is two times the sum of the focal lengths, f1 and f2, of the lenses required to separate and recombine the two halves of the input set. To maintain the same output channel spacing as that of the input, f2 must be twice f1. The total length is therefore 6f1.

Another implementation that has been proposed is one by Midwinter [53] which is suitable for one-sided operation, that is, the input and output elements are on the same side of the system. The advantage in this design is that the exchange switch logic array, which if not purely optical, can be on one side separated from the purely optical shuffle. The approach here is also, as in the previous scheme, to stretch-mask (shear)-add the inputs to obtain the perfect shuffle. However, the same optical system is folded in such a way as to incorporate a return path to the input side. The bottom half of the system only does a one-to-one imaging of the exchanged elements to the output port. Having the I/O on the same side is an advantage from the point of view of implementation. Figure 11.2 shows the folded perfect-shuffle scheme.

The third scheme by Eichmann [54], mentioned previously, is more compact than the previous two methods. Two versions have been proposed: one in which two identical negative cylindrical lens are used side by side, and another where only one negative cylindrical lens can be used with two prism wedges. In either case collimated input beams are required. The total path length for the scheme is 2f (for the first implementation).

11.1.2 Exchange switch

The focus by most researchers has been on the realization of the perfect shuffle connection. Since the shuffle connection can be done using passive elements, the system can operate essentially at optical bandwidths. The exchange switch for the SEN cannot be realized with simple passive devices since some form of control is required to either pass uninterrupted or deflect the input beams to the output of the switch. Unfortunately, scant work is evident on designing the exchange switch. We examine some alternatives in implementing a special case of the exchange switch as described by one reported work.

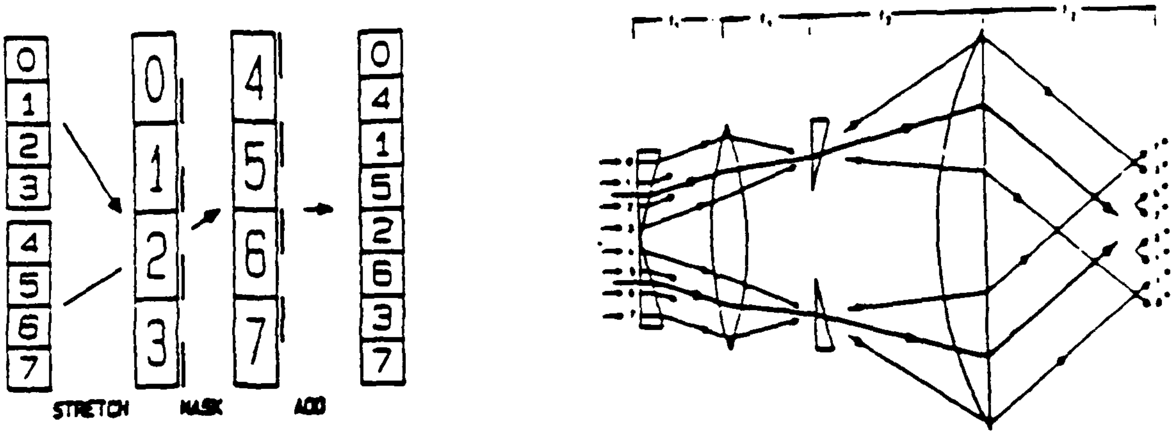


Figure 11.1 Lohmann's scheme for perfect shuffles

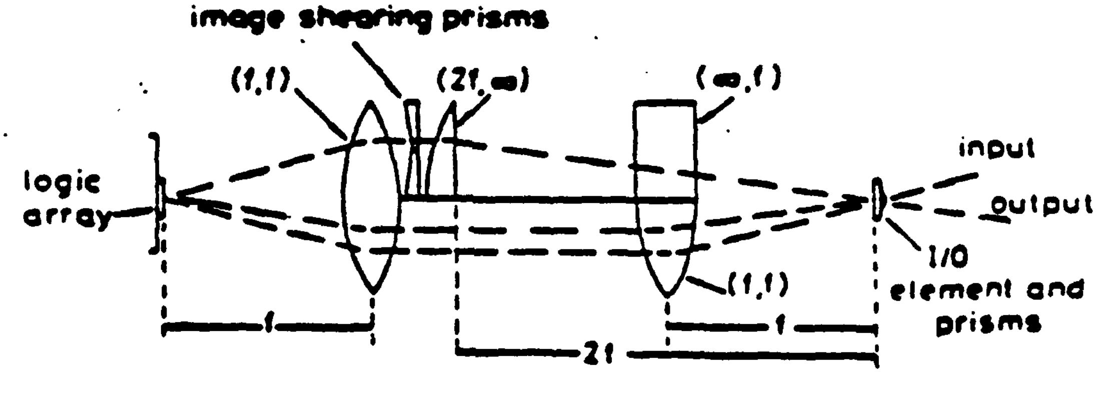


Figure 11.2 Reflected perfect shuffle

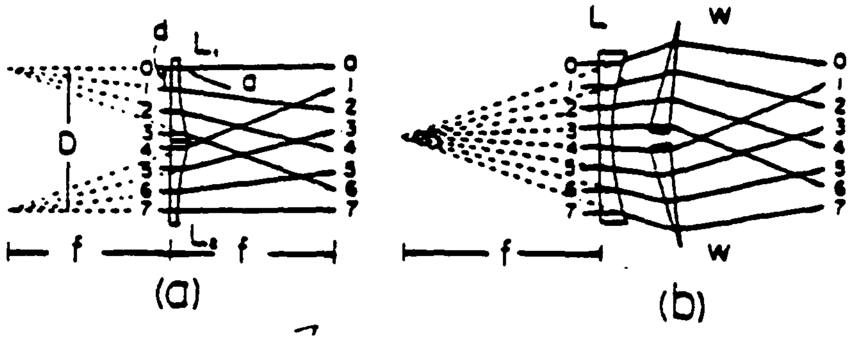


Figure 11.3 Compact perfect shuffle

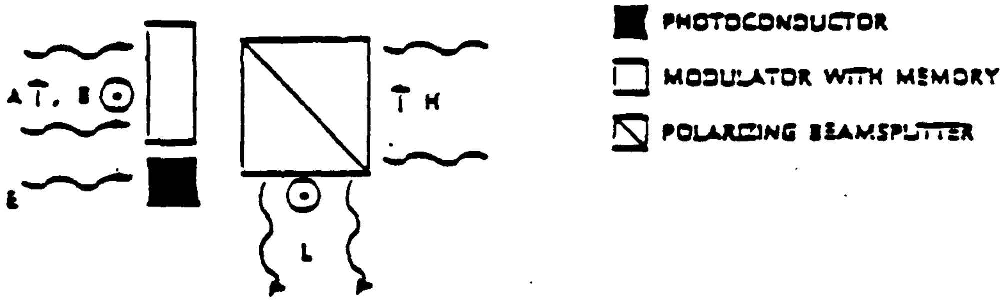


Figure 11.4 Compare and exchange switch

Stirk, et al, [55] examine means of constructing a compare and exchange module which is assumed to always receive two inputs. The exchange is based on comparison and not on any prioritized scheme as in the case of the message passing network of SPARO. However, some bulk optical techniques are discussed for realizing the basic exchange function, that is, the cross or bar (pass) configuration. Among the passive routing techniques suggested is polarization encoded switching using Wollaston prisms and controllable half-wave plates. The retardance of the half-wave plate is controlled electrooptically by some photoconductor. When the photoconductor is activated by the comparison signal, the dynamic half-wave plate rotates the polarization of the orthogonally polarized input signals through 90 degrees (see Figure 11.4). A polarizable beamsplitter or Wollaston prism can then spatially separate the two input signals. The performance of this scheme is bound by the bandwidth of the half-wave plate. These devices, due to their limited switching power dissipation, can respond at millisecond speeds. With newer ferroelectric liquid crystals, one can expect to push this response time to the microsecond range.

An electrooptic approach has also been suggested using a system of detectors and modulators. It uses nonlinear modulators which are normally transmitting unless a signal from the corresponding detectors converts them to be opaque. The scheme requires an electrical reset for the system to be operable on subsequent messages. Use of the electrical control thus requires using electrooptic devices such as PLZTs or ferroelectric liquid crystals, the performance of which would limit the message bandwidth.

In the above proposals, the response time of the control of the switch is always the limiting factor and offsets much of the speed advantages of the purely optical shuffle. (However, it is important to note that there are other advantages an optical shuffle can conceivably provide over an electronic one besides speed such as increased simplicity and physical compactness. We will visit these advantages later.) It is important to point out, that the above designs ignore many of the key functions that are essential to the exchange switch. A simple compare and exchange module will not be useful in the case of parallel processing that uses message passing. The next subsection summarily outlines the important functionalities required in the exchange switch as well as in the complete network implementation.

11.2 Required functionalities of the exchange switch

The survey of current attempts in designing optical SENs reveals that while the basic exchange switch has been examined, some important requirements of the exchange switch have been totally ignored. These requirements are necessary when the network is employed in a parallel processing environment as in SPARO.

11.2.1 Conflict resolution by the exchange switch

The most important problem that has to be considered in a message passing environment is that of the resolution of possible conflicts among messages that arrive simultaneously at an exchange switch. Conflicts between messages occur since the correct routing of both messages may require different settings of the exchange switch.

There are a number of approaches to address this conflict depending on the desired complexity of the switch. Since one message is going to lose the conflict, these approaches differ in how to treat the losing message. One approach is to drop the losing message and let the sender processor (the source of the message) wait for a response. If a response is not received within a specified time, the sender processor will retransmit the message. The processor thus follows a specific message protocol sequence. In the case of fine-grained processing, implementing a protocol or handshake with each message is too high an overhead and therefore not acceptable. In the second approach, the losing message is rerouted, that is, the message is deliberately passed through the wrong output but certain modifications are made so that it reaches its destination. In case of the single-stage SEN, the rerouting, consists of simply resetting the counter field that indicates the number of passes the message had made. Because of the way the counter is typically implemented in electronics, it will be referred to as the mask. The mask denotes the age of the message: in a network connecting N processors, a message is delivered in $\log_2 N$ passes when no conflicts occur. Thus to route a message, this mask is reset to 1 when the delivery is on mask value $\log_2 N$, usually represented modulo 1. No other modifications are necessary since the destination address does not change. The resolution of the conflict or the switch setting, as mentioned earlier, is done by selecting the message which has a higher mask (counter) value (prioritized selection).

11.2.2 Delivery of messages

A message is delivered when it has successfully cycled $\log_2 N$ times around the network. The delivery of the messages thus requires examining the mask or counter value. When the mask value has reaches $\log_2 N$, then it must be extracted from the network and delivered to the processor. This is envisioned to be simpler than comparing the destination address with the address of the processor associated with the output of each shuffle. Such a mask checking scheme would be especially attractive if the processors are operating electronically while the messages are optical signals. In this scheme, the mask of a message (or some optical equivalent) could be checked optically or electrooptically at every cycle without removing the message from the network system. When the message has indeed reached its destination, it can be sent out of the network, converted into an electronic signal and queued at the input message buffer of the processor.

11.2.3 Detection of a valid message

Because of the possibility of noise in an optical system, it is important that the network can distinguish a noisy null message from a valid message. Accepting noise as a real message can ruin the network performance by causing unnecessary conflicts at the exchange switches as well as send spurious data to the processors. The traditional electronic approach recommends providing a message header with each message. In the case of an optical implementation, one may either provide a header bit or stream or a separate signal which indicates if a valid message is present.

We assume a synchronous operation of the network so that it can work efficiently with electronic processors. A synchronous design will also be easier to design and implement. The network cycle will be synchronized with the processor array clock whose rate is determined by the speed of operation of the complete shuffle-exchange. At the beginning of each network cycle, the processors will be polled for messages generated for routing during the previous network cycle. The message present signal for every processor will therefore be examined during the beginning of every network cycle. Figure 11.5 shows the simplified schematic of the network and processor array interface. The message register in the network is required to hold a new message from the processor or a recirculating message from the network.

11.3 Possible approaches to an optical exchange switch design

The most difficult part of an optical SEN is the exchange switch design. Besides implementing the conflict resolution, the exchange switch design also governs the delivery of messages (since this depends on the way masks are represented and updated). Similarly, the optical technique used for implementing the basic exchange switch, that is, the simple cross or pass switch, also determines how the controls for conflict resolution will be realized. Our initial investigation indicates that the method of representing the mask information is critical to the nature of the switch design.

Here we discuss briefly the different candidate optical techniques that could be used for implementing the basic exchange switch. These are: acoustooptic gates, polarization encoding gates, waveguide or coupler, and photorefractive gates based on four-wave mixing. The goal is to pick the technique that results in the most speed-efficient basic exchange switch and then add the required functionalities of the network. We also provide one exchange switch design that we have investigated using Fredkin gates which have been proposed as an optical computing devices [56,57]. While no optical implementation of Fredkin gates are known, they can be viewed as a useful computing primitive from which complex computing structures can be built. The Fredkin gate design will be viable if the basic Fredkin gate can be implemented optically in a compact fashion.

11.3.1 Polarization encoding gate

The polarization encoding gate concept requires input message signals to encode the switching information as polarization. Thus, each message can have one of two polarization levels indicating whether the switch has to be in a pass or a cross configuration. The data in each message is assumed to be intensity-encoded. The optical switch is essentially a birefringent plate with a stored grating. The grating is visible to a message beam only if the message has the 'cross' polarization, in which case the input beam is diffracted across the plate. The situations when no conflicts occur, that is both messages require a cross or pass configuration of the switch, is relatively easy to implement. The cross configuration can be realized by allowing a negative diffraction for the upper beam and a positive diffraction for the lower beam. More details on this method are described in the next subsection on the final switch design.

Note that this approach differs significantly from the polarization switching gate discussed by Shamir et al [56]. In that gate, signals passing through a electrooptic modulator are rotated by 90° when the gate is activated electrically. The approach presented here is purely optical and appears to hold the most promise. However, the critical issue of conflict resolution can be incorporated in this encoding technique is not clear.

11.3.2 Acousto-optic gate

The acoustooptic gate is not very different from the polarization switching gate of [56], except that the switching information of the exchange gate is encoded in an acoustic signal. The gate is essentially an acoustooptic deflector that can be implemented in bulk or as an integrated SAW device. If the there is no acoustic signal on the gate control line, the input messages pass undeflected, otherwise they are deflected across. The problem with this approach is that the bandwidth of messages is in acoustic range which is lower than electronic bandwidths. While this is acceptable for transfer of large messages which arrive infrequently, it is too slow for messaging in a fine-grained computing environment where the rate of computing depends on the rate at which messages can be delivered.

11.3.3 Photorefractive gate

A photorefractive gate based on four-wave mixing is an all-optical approach mentioned by the authors in [56]. Besides the two incident input message beams, the control consists of two pump beams. The inputs are transmitted if the control is absent, otherwise they are phase-conjugated resulting in switching between the outputs. Given the state of art in four-wave mixing, this approach appears the least feasible for implementation.

11.3.4 Waveguide or coupler

A modulated waveguide coupler is used for the switch. By electrooptically changing the guided mode effective index, one can change the coupling between the two coupled waveguides. A number of techniques for implementing this function have been described in the last decade. More detail will be given in connection with the analysis of optical interconnection networks to be given later in the report.

11.3.5 Fredkin gate implementation

Fredkin gates have been proposed recently as building blocks for optical computing. As Figure 11.6 shows, the Fredkin gate is a controlled crossover device that can be used for constructing circuit primitives (such as crossover, fanout, and delay) and computing primitives (such as AND, OR, and NOT). A Fredkin gate is also a conservative logic gate [58], that is, it is reversible (information lossless) and bit-conservative (conserves the number of 1s and 0s that are present at the input). A control-specific Fredkin gate [57] is one in which the control and data lines are fundamentally different and cannot be interchanged.

We have investigated the use of Fredkin gate for realizing the exchange switch and its controls. Our approach has been to map a Boolean functional description into a circuit using Fredkin gates. We completed, as an example, a minimal Fredkin gate design [57] for the switch control. (At this stage we have ignored the mask update control.) We describe below the functional specification and the corresponding realization. The minimal circuit realization derived is control-specific with respect to the mask comparison information only.

We define five inputs to the exchange switch. These are:

P1: Presence signal for the upper input of the switch, indicating whether a message has arrived. A 1 indicates the presence of a message while a 0 indicates no message.

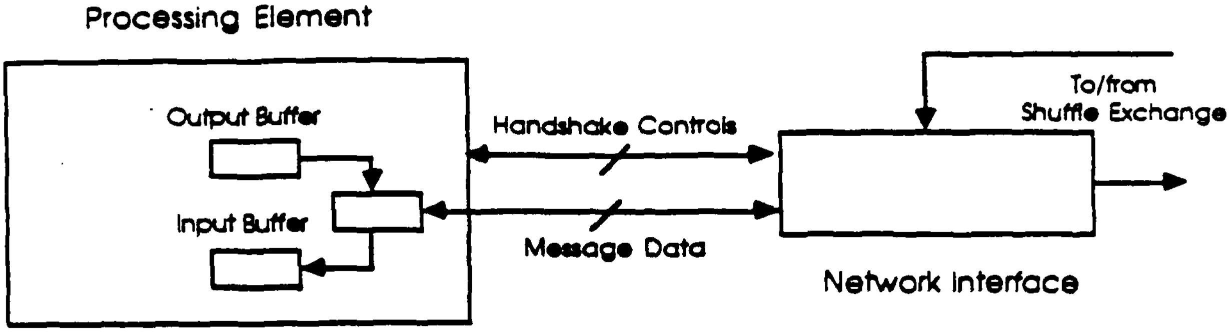


Figure 11.5 Schematic of network and processor array interface

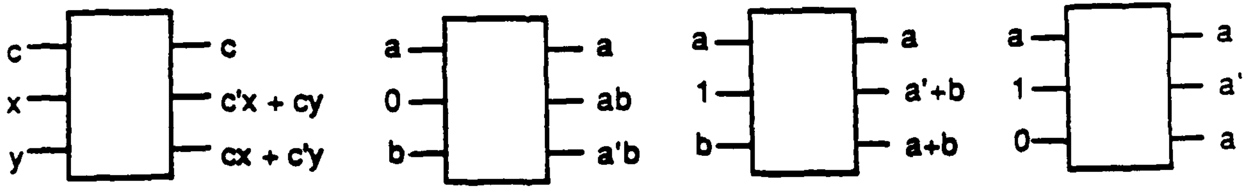


Figure 11.6 Fredkin gate and realisations of AND,OR,NOT.

		P_1	P_2	M	A_1	A_2	C	R_1	R_2
No Packets		0	0	-	-	-	-	0	0
One packet		1	0	-	0	-	0	0	0
		1	0	-	1	-	1	0	0
		0	1	-	-	0	1	0	0
		0	1	-	-	1	0	0	0
Two packets	No conflict	1	1	-	1	0	1	0	0
		1	1	-	0	1	0	0	0
Message Comparisons required	Conflict	1	1	1	0	0	0	0	1
		1	1	0	0	0	1	1	0
		1	1	1	1	1	1	0	1
		1	1	0	1	1	0	1	0

Table 11.1 Lookup table for smart exchange switch

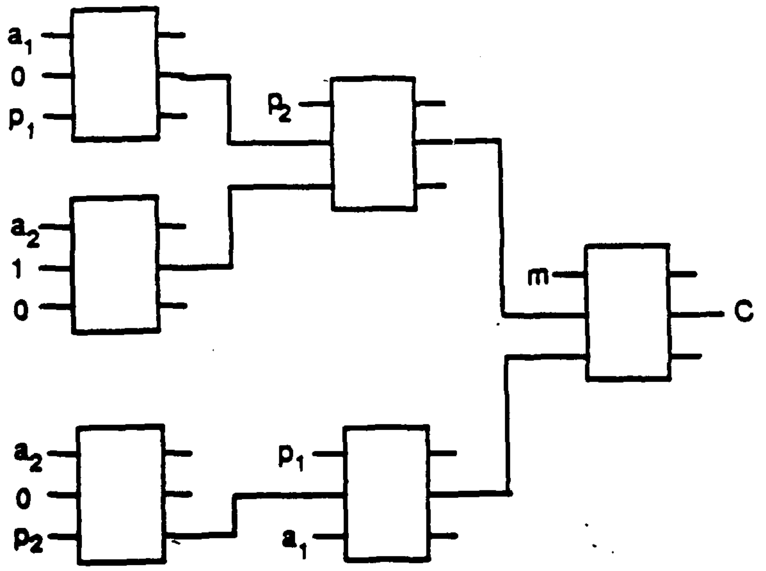


Figure 11.7 Exchange switch implementation from Fredkin gates

P2: Presence signal for the lower input of the switch.

M: Mask comparison signal, or the outcome of the comparison $M1 <= M2$ where M1 and M2 are mask values of the upper and lower input messages, respectively.

A1: The destination address bit under the mask M. This bit decides the switch configuration required for the message in the current network cycle.

A2: The destination address bit under the mask M2.

C: The output of the Fredkin gate circuit which represents the control input to the last Fredkin gate that acts as the basic exchange switch. A C value of 0 implies a straight or pass configuration while a value of 1 represents an exchange or cross configuration.

The truth table for setting the control C is shown in Table 11.1. Note that R1 and R2 in the truth table represent the mask reset controls that are necessary to handle conflicts. A logic minimization of the combinational function for C yields the following sum of products form expression. A _ before a variable name denotes the complement of that variable.

$$C = P1 -P2\ A1 + -P1\ P2\ -A2 + P1\ A1\ -A2 + P2\ -A2\ -M + P1\ A1\ M$$

The expressions for the reset signals are:

$$R1 = P1\ P2\ -M\ (-A1\ -A2 + A1\ A2)$$

$$R2 = P1\ P2\ M\ (-A1\ -A2 + A1\ A2)$$

Figure 11.7 shows a six gate implementation for the exchange switch in terms of the five input variables. The total delay in this switch is three Fredkin gate delays.

As indicated earlier, the feasibility of the above implementation depends on the feasibility of the Fredkin gate implementation. As yet a feasible optical Fredkin gate has not been demonstrated. There have been some discussions [19] of cascading Priese gates or interaction gates to construct a single Fredkin gate but they have not been explored in detail. The speed and complexity issues of the our design will determine whether such an implementation is viable.

We now present a conceptual optical switch design based on the study of the methods described here. The results of this effort indicates that while parts of the exchange function for the switch can be implemented very elegantly using optical computing techniques, the optical implementation of the entire exchange function and its control is beyond the feasibility for current demonstratiͻn.

11.4 Design and analysis of an optical exchange switch

The basic exchange switch is based on the polarization coding method described previously.

11.4.1. A passive optical exchange using polarization as control.

As described earlier, passive exchange of signals in a SEN is possible by forming a grating structure in a birefringent material so that light propagating with one polarization is affected by the grating and deflected to the appropriate path, while light of the crossed polarization is unaffected by the grating and passes through the material essentially undeflected.

To operate such a polarization-based switch, the polarization of the light beams (messages) must be switched or set based on the switch setting control logic that is a function of the address bits and the mask values or the message ages. (Note that in case of an optical implementation, we will use message ages to denote mask values since the latter has an electronic implementation.) While many devices are available which can switch polarization quickly based on electrical control [59,60], all-optical devices tend to perform this operation either slowly or over a relatively long interaction distance. While the actual routing is passive and optical, the control for this polarization setting operation must be electronic if it is to be implemented with components and materials existing at this time.

One advantage of this approach is that the information carried along with the message regarding whether an exchange should occur or not does not have to be actively decoded at the routing device. Instead, when this information is calculated, it can be represented in such a way that the decoding is a passive beam propagation phenomenon. Since the calculation of whether a particular message should be exchanged or not must be performed at each step through the network anyway, the overall switching time can be reduced.

11.4.2. A passive optical message age update.

In the SEN, in addition to address information which conveys to a switching node whether it should exchange inputs or not, a message also carries its age information. Again, the large number of information representation formats for optical computing can be used to reduce the overall speed of updating this information. Since the propagation of information through the network can be thought of as routing in a plane (yz in Figure 11.8), in real spatial dimensions, an optical routing implementation might be performed with planar geometry, using mirrors, prisms or gratings to deflect the paths of the beams carrying the messages through the network. This leaves another dimension (x) orthogonal to this plane to use as information encoding. One simple method of representing and updating the number of passes a particular message has taken through the network is to represent the number of passes as a spatial position in this orthogonal direction. With each pass through the network, the position of the message beam is shifted one unit in the perfect shuffle plane by means of a prism or a grating. Because this information is represented spatially in a dimension (x) orthogonal to the interconnection plane, shifts in this dimension can be independent of the routing pattern in the interconnection plane. Since there is no decision on whether to shift or not because propagation of any message for another pass represents a shift, this can be performed with a permanently configured passive optical system of gratings or prisms.

11.4.3. Detection of age and collision decisions

Perhaps the biggest shortcoming of the all-optical exchange-switch is the lack of a method or devices to perform the decision on which of the two messages gets priority of the exchange control when a collision occurs at a node in the network. This decision is based on the ages of the two messages, usually with the older message getting priority and the younger message starting routing afresh in the network.

Optical computing strategies have traditionally had serious problems implementing integer comparisons. One method that could accomplish this with light-speed throughput is a look-up table hologram in which all combinations of age comparisons are stored, and the appropriate combination is recalled through Bragg reconstruction of the correct output. However, the number of combinations which the hologram would need to store for a large system would be prohibitive. Referring to the last four rows of truth table shown in Table 11.1, one can verify that the number of entries in the truth table is equal to the order of the number of message comparisons or $O((\log_2 N)^2)$ for a SEN connecting N processors (PEs). The actual number of entries required is $4(\log_2 N)^2+5$. Thus for a SEN connecting 1K or 1024 PEs the look-up table required for every exchange switch control must have stored in it 405 entries. The number of inputs is 24, while the number of outputs (control of switch configuration, reset controls for the age of each message) is 3. A single hologram of this size is not a problem. But since N/2 exchange switches are required, the 1K network must be able to physically accommodate 512 such holograms.

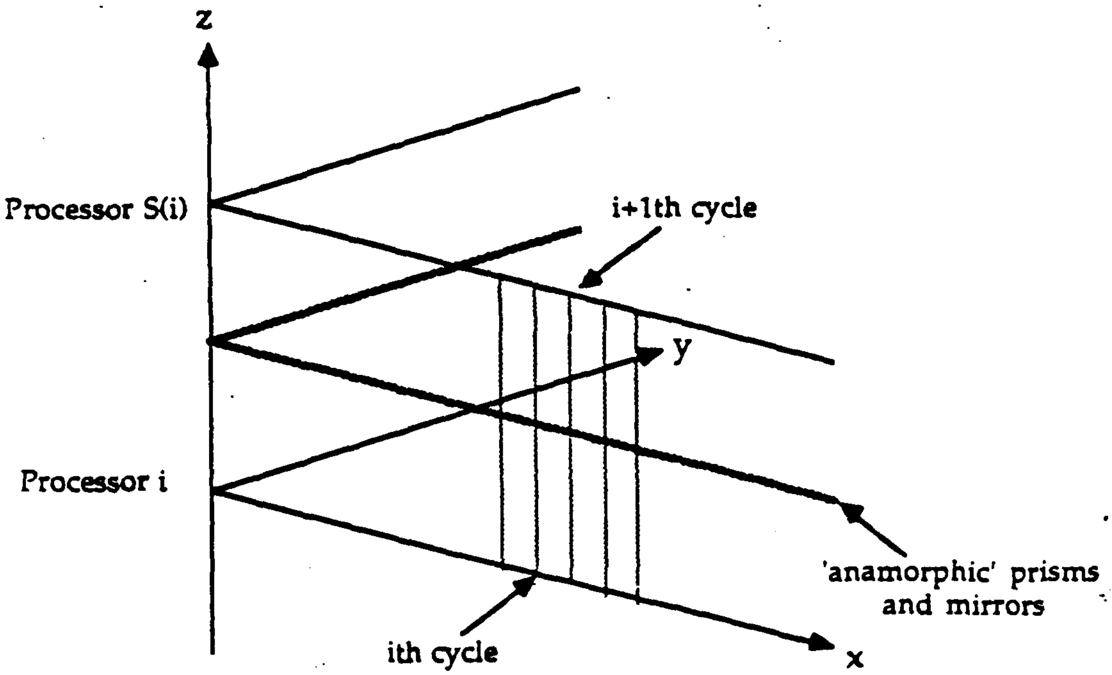


Figure 11.8 Representation of message age by spatial encoding

Other possible approaches would require either the representation of the ages or the number of passes as analog levels and a threshold comparison of the two intensities or a Boolean algebra calculation of the comparison. While fast analog threshold detection might be possible with multiple quantum-well devices, the fabrication technology of these devices is not yet advanced enough for them to be used reliably in large-scale network implementations. Boolean operations for optical computing are also not feasible for large systems at this time. Devices which perform fast Boolean logic operations in optics tend to be multiple quantum-well devices, which are difficult to use for large systems. Spatial light modulators, which can be easily expanded to perform Boolean operations for large systems, unfortunately operate at very low speeds.

11.4.4 Conclusions in designing an optical smart exchange switch

In summary, there are two aspects of the exchange operation in which optical implementation can be applied to decrease routing delay time for messages passing through a SEN. These are the passive exchange of polarization coded messages, and the passive update of the number of passes that a message has taken through the network on the way to its destination. Unfortunately, this is not enough to justify an all-optical implementation of exchange operation, because the control of polarization changes and the decisions for handling collisions cannot be implemented easily in high-speed devices with optical control using components which are available at this time, or which are even in developmental stages in any laboratory.

11.5. Hybrid SEN Implementations

To quantitatively evaluate the relative advantages of an optical SEN, specifically the optical shuffle, we examined two different SEN implementations. The first implementation is a purely electronic design of both the exchange switch and the shuffle connection. A logic design and analysis was conducted to determine the complexity of the design in terms of silicon area and wire lengths required for connections. The same analyses also yields the first-order estimate of the speed of operation of the SEN when the fastest semiconductor technology is used. The second implementation is a hybrid one that uses an optical shuffle in conjunction with an electronic exchange switch. The optical shuffle section is expected to be slightly faster and much more compact than a hardwired electronic implementation. A similar analysis as in the case of the first was undertaken to assess the second design. An optical SEN implementation can then be compared to both these implementations on the basis of their speed-complexity product.

11.5.1 SEN and processor array interface

The single-stage SEN was shown in Figure 10.5 showing the Processor Array, consisting of 1024 processing elements (PEs) for fine-grained computing, communicating with the network through the Network Interface (NI), or the control portion of the network that handles the transfer of messages between the PEs and the network. Here we will focus on the complexity and the cycle time, the delay experienced by a message to pass once through the shuffle-exchange stage, of the network.

To isolate the performance of the SEN from that of the implementation of the PEs, we will define the network cycle to be the difference of the time when a message is accepted into the NI and the time when it is loaded back into NI for delivery or for recirculation.

Each PE, as Figure 10.5 shows, contains two buffers for storing outgoing and ingoing messages. A message to be sent to another PE is queued at the output buffer (OB). At the beginning of a network cycle (defined as the time taken by a message to cycle once through the shuffle-exchange network), if there is a message in the OB, the PE requests access into the network through the handshake line Processor Request. The network can receive a message if there is no circulating message at the PE. The NI corresponding to the PE communicates this information to the PE via the Processor Access line. When access is granted to the PE, the message is loaded from the OB into the NI through the message data line/lines. Similarly, when the NI at the end of a network cycle has a message to be delivered to the PE, it uses the handshake signal Network Request to check if the Input Buffer (IB) of the PE is not full. If IB is not full, the PE uses the Network Access line to signal the NI to transfer the message over the data lines. Note in Figure 3.5 we have assumed that the data lines between the PEs and the NI are bidirectional ports. This has been done to reduce the number of I/O connections.

The mechanism for delivering messages from the SEN to the PEs proceeds as follows. When a circulating message is received in the NI, it is checked to see if it has completed $\log_2 N$ passes successfully. If it has, then it is transferred to the PE, otherwise it reenters the SEN. This checking can be done within the NI module or inside the SE stage.

We now focus on the Shuffle-Exchange (SE) part of the network independent of the network interface portion.

11.5.2. Shuffle Exchange Stage

The basic exchange switch, shown in figure 11.9, is one that is controlled by one input that produces a cross or bar connection between the input and the output. An useful exchange switch must satisfy all functional requirements listed earlier. We will call such a switch the smart exchange switch (SES) described by the Boolean equation. The Boolean expression for the switch control assumes that both the deciding address bit extraction as well as the mask comparison has already been completed. There are a number of possible hardware schemes, serial and parallel, for realizing both operations. We now consider some electronic implementations.

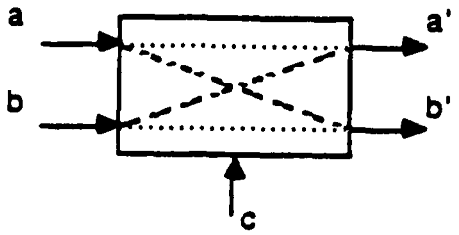
11.5.3. Electronic SEN Implementations

In order to uncover the critical architectural issues in designing optical SENs, we first consider a purely electronic design. A number of issues determine the nature of the electronic implementation in terms of the size and complexity as well as performance. These include pin constraints of a chip, the number of backplane interconnects between printed circuit boards or the backplane wiring constraints, and the off-chip and off-board interconnect delay as compared to the gate delays within a chip. We examine these issues briefly as inputs to our design methodology, and then present the possible performance and size of electronic implementations.

11.5.3.1 Pin Constraints

The design approach most affected by the limits on pin counts is the issue of parallel versus serial message data transfer. Consider the parameters of the SPARO (Symbolic Processing Architecture in Optics) architecture. There are nominally 1024 PEs communicating via messages composed of five essential fields: the destination PE address, the source PE address, two data/address operands, and the instruction. For a 1024 processor architecture, the address is 10 bits wide. Although the specific application will decide the size of the data used, we assume that a 32-bit wide data would suffice. The instruction word was assumed earlier to be encoded as 1 bit per macro-instruction to simplify the instruction decoding in optics. We expect that more than 30 macroinstructions, each of which requires 2 to 3 machine cycles. In the case of an electronic design where binary decoding can be done relatively simply in the PE, we will assume 6 bits are required to encode all macro-instructions.

The total width of the message is thus 90 bits (2(32) + 2(10) + 6). A message of this length presupposes that the mask for each message is generated in the network. However, to avoid increasing the complexity of network, it is preferable that the mask be generated by the processor and sent as part of the message for purposes of routing in the SEN. The total message length would then be 100 bits. Larger or smaller sizes of the messages are possible depending on the size of the data necessary. As we shall see later,



$$a' = \text{NOT}(c)a + cb$$

$$b' = ca + \text{NOT}(c)b$$

Figure 11.9 Basic exchange switch

the length of the message has a profound effect on the performance and implementation of the SEN.

The control lines required between each PE and the corresponding row of the SE were discussed earlier. Four handshake lines are required: Processor Request by the PE to transfer a message from the OB of PE into the NI, Processor Grant to grant the processor request, Network Request by the network for message delivery from the NI into IB of PE, and Network Access to grant the network request. Since messages are all of fixed length, we do not require acknowledge signals after message transfers have been completed.

The problem of pin limitations arises when considering how the message has to be transferred between the PEs and the SEN: 104 bits of control and message information if the messages are transferred in parallel between the NI and the PE, or 5 bits for control and serial data lines. Note that in our initial electronic designs all message lines are considered to be bidirectional to reduce the space overhead. In the first case, all messages (at the beginning or at the end of the network cycle) can be transferred in one clock cycle after handshaking. In the second case, 100 clock cycles are required to transfer the message serially. While the serial option is 100 times slower, it requires 1/21th the number of pins at the output of the PE. There is thus a space-time trade off to be considered. The real question to be answered is the total space-time complexity. If 1024 PEs are placed on one board to reduce off-board delays, then the board has to accommodate 1024*104 or 106496 lines between the PE array and the SEN. By comparison, the serial transfer scheme only requires 5120 lines. How many PEs can be put on a chip or package and then on a board therefore depends on the pin limitations on the chip as well as the number of interconnection lines that can be squeezed on a single board. These issues in turn depend on the technology used to design the board.

11.5.4. Off-chip Interconnection Delays

Since the complete network, that is, the exchange circuitry as well as the shuffle connection, requires a multi-chip (possibly multi-board) implementation, off-chip delays will be a design concern. The problem of interconnection delay becomes severe for a large perfect shuffle where exchange stages are switched at high speeds. The network delay, or the time taken by a message bit to pass around the complete SEN, depends partially on the actual interconnection length between the output of the exchange stage and the register that delivers the message to the input of the exchange stage. Since the shuffle connection is not modular, this length increases with the size of the network. We will examine the relative importance of the interconnection delay when examining SENs implemented in different electronic technologies.

11.5.5 Backplane Wiring

The problem of backplane wiring arises if a multiboard implementation is necessary when all PEs and the corresponding interconnects cannot fit on one board. The number of boards and the total delay would then be determined by the number of backplane interconnects possible. The maximum number of board-level interconnects depends on the technology used to construct the board. Thus, using thin film multilayer (TFML) boards allow for much faster and more dense interconnects than do standard PVC PCBs with edge connectors.

Both pin count and packaging constraints and limits on backplane wiring will therefore determine the nature of transfer of the message, that is, how serial or how parallel. The other factors are the complexity of each SE stage. The larger the area of a single SE stage, the fewer PEs can be fitted on a chip and thus fewer chips on a single board. We will consider these factors in more detail in the next section.

11.5.6 High-Speed Electronic Implementation

We have examined the complexity of the circuitry required to implement the complete shuffle-exchange network in electronics. Both GaAs and Si ECL technologies were considered for high-speed implementations. Given the large size (1024) of the network, we initially considered bit serial transfer of messages. We examine the design implications for a parallel message transfer scheme later. As we will show, the nature of the message transfer in the network is critical in determining the complexity and performance of the network implementation.

The size of each exchange stage and its controls is estimated first to determine the layout complexity and thus the total area, size, and speed of the network. The circuitry in the exchange includes the NI and its controls, the combinational logic to generate the exchange switch settings, and registers to hold the message during recirculation. Figure 11.10 shows a schematic of the electronic SEN. The operation of the network is now explained in more detail.

A message is accepted into the network with the destination address field entering first. The address and mask fields of the message are successively loaded into their respective registers, so that the deciding address bit can be extracted. The deciding address bit extraction is achieved by serially shifting the mask and address register contents and ANDing the output bits. While the mask register is being cyclically shifted, the mask comparison between masks of two messages can be done in parallel. The mask comparison result and the deciding address bits are fed to the switch and mask control logic. While the presence bits of messages

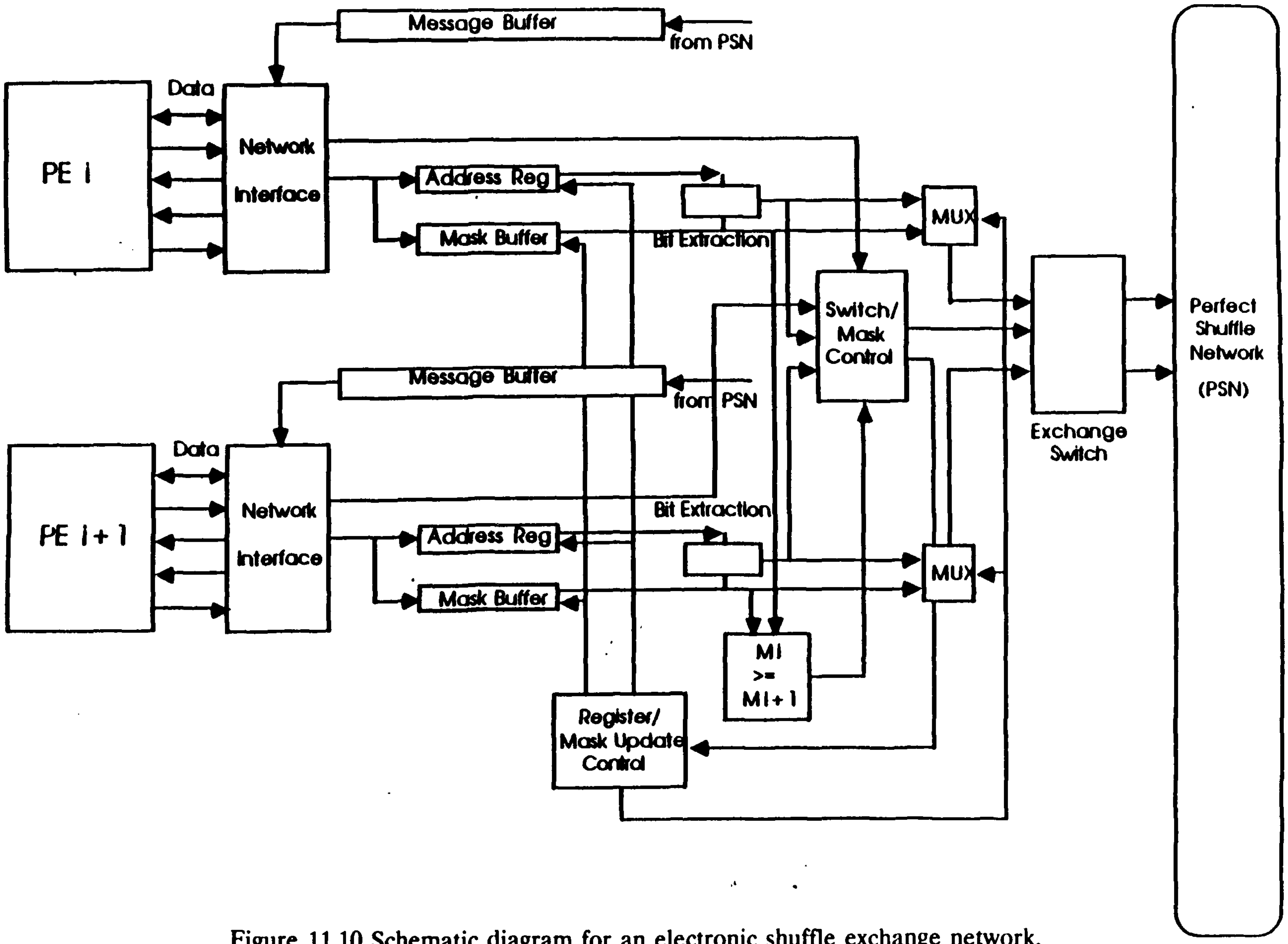


Figure 11.10 Schematic diagram for an electronic shuffle exchange network.

have not been shown they can be derived in the NI by examining the processor request and grant lines. When the exchange switch is set, the registers are alternately emptied to serially pass their message to the shuffle stage. The message buffer holds the initial portion of the message while the remaining message portion passes through the exchange switch.

We now examine implementation of the above SE circuit in different technologies.

11.5.7 Network delay for GaAs implementation

In the case of GaAs, the total number of gates (the average gate is a two-input NOR or NAND) required for each exchange stage for every pair of PEs is expected to be less than 1300. (The major proportion of these gates are consumed by registers and buffers.) If the fastest usable GaAs technology were employed, gates with delays of the order of 100 ps (10 GHz) could be used. Since the number levels of logic between switchable gates will be 3 to 4 on an average, the clock speed for operating the network would be about 2 GHz. At that speed, the current and near-future level of integration allows between 3000 and 4000 gates on one chip. This allows us to realistically fit in 2 exchange stages (for 4 PES) on a single chip. Using state-of-the-art multichip carriers, we could fit close to 25 chips or 50 exchange stages on a rectangular multichip carrier package (MCP) measuring 3.7 by 2.4 inches. The current limit on the number of pins in a pin grid array is about 575. Since each stage requires one output message line and five input lines, a single package can be used to pack about 100 PEs. We are assuming that the data line is bidirectional. If unidirectional data lines are used, six lines are required to connect each PE to the network.

For a 1024 SEN, we would have to place at least 10 or 11 such MCPs on a PWB (printed wire board). Since standard PWBs have low dielectric constants, off-package delays will be higher than those inside the package unless polyimide substrate is used for the board. Interconnection lines would be done in copper on the polyimide. The shuffle stage would have to interconnect these 10 packages on the board. Each package would have 100 serial output lines for messages leaving the exchange stage for the shuffle connection.

Because of the non-local nature of the shuffle connection and the limit in integration, we cannot integrate the shuffle connections on the package. The off-package delays for the connections are directly proportional to the wire length. If multilayer (TFML) connections are used (currently 5 layers with 3 for ground and power), the longest vertical length between packages is 5 MCP heights, that is, 5" x 3.7" or 18.5". Since this line is large in length, there is considerable loss in the lines. It would be difficult to run GaAs at 2 GHz over long interconnection lines unless impedance matched input and output buffers are used at the input and output pins of the exchange stages. (The bigger problem is that of timing, or the distribution of clock to all parts of the SEN which is to operate synchronously.) Since the delay on copper on polyimide is 62 ps/cm, the longest interconnect delay in the shuffle stage is 2913.4 ps or nearly 3 ns. Note that the processor to exchange stage connections are not as much of a bottleneck since they are direct local connections between PEs and the MCPs for the exchange. We assumed, in our preliminary analysis, that all PEs can be connected by the SEN on one board. As it turns out, this is not possible since packing 1024 PEs alone will consume a complete board-see note on processor complexity in the next subsection. Thus the PE array and SEN connection will be across boards and not on a single board. For purposes of determining the upper limit on the SEN performance, however, we assume that the PE and SEN connection problem can be solved, and therefore focus on the one-board SEN performance. We have assumed that load impedances will be matched at the MCP pin boundaries. Thus the delay between connections of successive stages of the network is over 3 ns given that some gate delays have be accounted for within the package. We can assume the worst case total network delay time to be about 4 ns.

In our delay computation, we have assumed that the clock can be distributed such that no clock skews occur. At Honeywell we use a star configuration in distributing the clock within a package to ensure that the clock propagation delay is the same for all modules. If clock synchronization is a problem at the board level, and we believe it will be, we could conceivably employ a holographic optical element (HOE) to distribute the clock.

11.5.8 Network Delay for ECL Implementation

Because of the relative severity of off-chip delays in the GaAs implementation, we considered a more mature technology, ECL, as another choice for implementing the control and exchange stage of the network. The advantages of ECL over GaAs despite its slower speed is the higher level of integration as well as a relatively smaller penalty for off-chip connections. Today, we can integrate 10000 gates on an off-shelf ECL gate-array chip with relative ease. With the state-of-the-art ECL technology, maybe even 20000 gates could be put on one chip. (This would be possible since the connections in the SE stages are regular and simple with low fanouts.) We are assuming of course that the high heat dissipation problem for the on-board ECL circuitry can be solved. With such a level of integration, one could account for SE stages for 20 PEs. To completely fit all 1024 PE connections, we would need about 50 chips on a board. Each chip would have 200 pins for serially transferring data in and out of the network. This number of pins is quite feasible today. By pushing technology to its limits, the data could be clocked through at 200 MHz if delays across the board are not significant. It would be reasonable to assume that the interconnection length would be close to that of the GaAs SEN. Thus the network delay will also be close to 4 ns. We assume as before that an optical clock distribution is possible using a HOE (holographic optical interconnect) scheme. Since the ECL clock is run as much as 50 times slower than that for GaAs, the interconnection delay in the ECL SEN is a very small fraction of the cycle time.

We now examine the total cycle time of the network for each technology.

11.5.9 Network Cycle Time

The network cycle time is the sum of the propagation delays for the control and the exchange switch, the shuffle path delay, and the total time to transfer the message. Since the number of gates is expected to be less than 5 between clocked stages, and since the shuffle connections are compact, these delays do not slow the clock down. The message transfer is therefore dominated by the time taken to transfer 100 bits. Since the delay across the board is not significant, the head of the message will arrive through the shuffle connection to the next exchange stage before the tail of the message has passed completely through the exchange switch. Even if the two 10-bit registers are used to alternately hold 10-bit chunks of the message, a message buffer is required to save a portion of the message before the exchange switches can be set for the next pass.

In the case of GaAs, the message bits are pipelined out of the source PEs at 2 GHz. The network cycle time is 54 (100*500 ps + 4 ns) ns or effectively the network operates at 18.52 MHz. The network delay time is thus 108 times slower than the GaAs gate delay (500 ps) because of the serial transmission of the message. Since two 10-bit registers are used to hold portions of the message, a 80-bit message register is required. In case of ECL, the message transfer time is 504 (5 * 100 + 4 ns) ns. This implies that the network effectively operates at 2 MHz.

For either technology, the serial message transfer causes the network to be a bottleneck. This bottleneck is especially serious when the PEs operate on the same clock as the SEN. An average message requires $1.5*\log_2 N$ network clock [46] or $150*\log_2 N$ clock cycles to be delivered (100 bits per message), when the network load is not more than 0.25. Therefore for a 1024 SEN, a message requires 1500 clock cycles to be delivered. If a specialized reduced instruction set (for combinator graph reduction) architecture is used to implement the PE, a message can be generated at best in 3 to 5 cycles, assuming some parallel loads are allowed within the PE. Thus, the message generation rate (all 100 bits generated in parallel) is about 300 to 500 times higher than the message delivery rate if the message is transferred serially, and 3 to 5 times higher if transferred in parallel. In an ideal situation, where no bottlenecks exist, the network should deliver messages at approximately the same rate at which the PEs generate them. Thus, the message bandwidth in the network must equal the message generation bandwidth in the PE. This implies that the SEN operates on a clock that is either operating at a ridiculous 300 to 500 times faster than the PE clock when transferring messages serially, or accepts and transfers messages totally in parallel with a clock that is less than an order of magnitude faster.

11.5.10 Parallel versus Serial message transfers

The previous discussion showed that the message bandwidth in the network is inversely proportional to the message length. The increased bandwidth required in the SEN motivates us examine the complexity of a parallel implementation where messages are transferred in a parallel or quasi-parallel fashion.

For the same level of integration as described earlier, an ECL chip built with 20000 gates can accommodate exchange stages for 20 PEs. However, instead of 20*6 (5 between PE and exchange and 1 between output of the shuffle stage and exchange) or 120 I/O pins, now 20 * 204 (104 between PEs and exchange and 100 between output of the shuffle stage and exchange) or 4080 I/O pins are required, ignoring ground, power and clock connection pins. Since this is not possible, a single chip cannot be deemed to contain more than one exchange stage for 2 PEs since about 250 is the limit to the number of pins to a chip. In such a case, 500 such chips would have to be interconnected in a shuffle connection where each channel of the shuffle now has to connect 100 wires per channel or a total of 102400 wires for a 1024 shuffle. If board-level interconnects are used, then many boards are required in the implementation of the shuffle connection since typically only 250 to 300 backplane connections are possible with standard edge connectors. Clearly, the pin limitations and size complexity makes a large shuffle-exchange for parallel message transfer impractical in electronics.

In summary, one notes that an electronic implementation, GaAs or ECL, for a large shuffle-exchange could be operated at high speeds, over 200 MHz for ECL and over 1 GHz for GaAs. However, severe limitations of the packaging technology and the level of integration of high-speed semiconductor technology forces a serial transfer of messages when a large SEN is desired. Unfortunately, when messages are transferred serially, the message throughput varies inversely as its length. For a modest message length of 100 bits, suitable for the level of fine-grained computing, the message throughput is less than 1/100th the data rate. Table 11.2 summarizes the electronic SE network cycle times and the corresponding message latencies. The message latency is defined, for our purposes, to be the average time required to deliver a message. We have assumed, using Lawrie and Padua's results [46] that a message requires an average of $1.5\log_2 N$ cycles to deliver

Technology	Gate delay	Network latency	Message latency
GaAs	500 ps (2 GHz)	54 ns (18.52 MHz)	810 ns (1.24 MHz)
ECL	5 ns (200 MHz)	504 ns (2.0 MHz)	7.56 us (132.3 KHz)

Table 11.2 Performance of electronic SENs.

if the network is not loaded much beyond 0.25 (that is, about 25% of all stages have messages in transit).

For comparison with the above performance, we examined the same issues for electrooptic and optical implementations.

11.6 Optical and electro-optical SENs

As in the electronic implementation both parallel and serial message transfers can be considered. As before a parallel message transfer scheme in optics requires that each channel be 104 (100 message lines and 4 control lines) signals wide. A total of 100K signals in a single optical system appears difficult in the near future if guided optics is used. Thus, the optical system may have to be quasi-parallel since the worst case serial transfer cannot provide acceptable throughput unless the switching speeds in the optical SEN is hundreds of times faster than the PEs.

Since an all-optical implementation has not yet been designed, we will focus on the architecture of electrooptic and hybrid implementations. One thing is clear: if the optical network has to provide an advantage over an electronic one, using an optical serial shuffle together with an electronic exchange and control will not be an advantage. This is because the serial delay of the message in electronics is a serious bottleneck, and using an optical shuffle will only add further electron-photon conversion delays. We will therefore examine optical methods to improve the bandwidth of message transfers. Another issue that requires consideration in designing the SEN is the nature of its interface to the PE array. The network interface depends on the size of the PE and therefore on its complexity. The next subsection examines the complexity of a special-purpose graph reduction processor.

11.6.1 Processor Complexity

The nature of optical implementation depends on the level of connectivity, that is, whether the connections between the PEs are within the board or off-board. Since we are building a fine-grained parallel system, the size of each PE dictates the nature of connection. For this purpose, we examined the functional requirements and the architecture of a specialized combinator graph reduction (CGR) PE.

Our initial estimates show that a reduced instruction CGR PE will have about 30 to 35 hardwired instructions, each of which executes 2 to 3 steps. The typical sequence of operations are as follows. The PE receives a message, decodes it, operates on it, and sends out a message in response. If a simple ALU (no multiplier) is used, the PE could be implemented with about 1600 gates in Honeywell's high-speed (50 MHz), high dens:ty CMOS process. To give more power to the PEs, a larger ALU equipped with a multiplier could be shared by a pair or more PEs on the same chip. As many as 11,000 gates can be put on 1 CMOS gate array chip (400 mils X 400 mils). Thus, 6 to 7 PEs can fit on 1 chip. As many as 20 gate array chips could be squeezed on a package 3.25 " X 2.6". Thus one package would account for 120 to 140 PEs. The limit in the packaging is not in the total gates available but in the number of I/O pins. The maximum number of pins possible in such a package is 575 in a pin grid array. Thus if 100 PEs were dedicated to a package, each would have only 5 or 6 (depending on a bidirectional or unidirectional data line) I/O pins allocated to it, ignoring common clock, power, and ground lines. This limited pin allocation per PE forces a serial message transfer scheme wherein all messages in and out of the PE have to be serial.

At the level of integration discussed, at most 10 packages can be fit on a large board. The limit at the board level would be in the number of board connections as well. If 1024 PEs were accommodated on 1 board, the number of I/O lines in the board at 5 or 6 I/O pins/PE will be over 5000. Thus, connecting a SEN to this board is not possible using standard electronic wiring connections. More importantly, it is clear that to build a scalable machine, consisting of 10,000 PEs or more (necessary for fine-grained computing), a multiboard solution is desired. Therefore, the SEN must be operational across multiple boards and not just within the board. The number of boards required for all PEs is dependent on the functionality and granularity of a PE. We now examine this important issue in more detail.

11.6.2 Processor Granularity

The PE granularity influences the SEN design in two ways. First, the coarser the granularity, the fewer the PEs required to solve the problem. This implies that a small number of boards will suffice to accommodate all PEs. The architecture of the processor/memory subsystem for coarse-grained processors will of course be quite different from the one chosen here. Second, coarser-grained PEs will operate on a larger problem (that is, on subgraphs rather than on individual nodes in CGR) and therefore will have less frequent communication with other PEs. The message generation frequency will therefore be considerably lower. However, the size of messages may be considerably larger. For example, the messages may contain subgraphs rather than single node information. The increased size of messages will tend to keep the bandwidth of messages high even if the message generation rate is decreased. One way to keep the message size down to the lengths that we are considering here (100 bits) is to use a radically different architecture such as shared memory and PE clusters. Using such a different architecture implies solving a different sort of PE communication problem which we will not consider here. We will instead focus on the general tradeoff of PE granularity and message bandwidth.

In the SPARO architecture that has been developed for fine-grained CGR, each PE contains and operates on a single graph node. There is no concept of memory since the registers in the PE specify a graph node completely. While this approach seemed suitable for optics where a complex processor could not be designed, when considering an electronic implementation other problems surface. First, the number of PEs required in the architecture is not defined by the size, in terms of the number of nodes, of the original combinator graph, but by the maximum number of nodes required during reduction. As recursive expansion of functions is common in CGR, we expect that the maximum number of PEs/graph nodes required for a real application may be as high as 100K to 500K, even when concurrent distributed garbage collection is employed as in SPARO. The maximum size of the PE array thus can be very large. For this reason, we may consider a couple of options when implementing SPARO realistically in electronics. As suggested above, we can increase the granularity of the PEs to handle subgraphs instead of single nodes, so that 1K PEs would suffice in handling reductions. However, this would reduce the maximum parallelism that can be expressed in the graph. It would also increase the memory requirements in each processor. As mentioned earlier, the message bandwidth is not expected to change much from that in SPARO since the messages will be of greater length but they may be generated less frequently.

For purposes of solving the PE interconnection problem, we can still derive a major benefit by solving the general message passing problem that features the same bandwidth as that of the messages in SPARO. We will therefore isolate the exact processing nature in the architecture used from the specification and requirements of the network, and focus on achieving a high throughput of messages between PEs in a generic parallel processing environment that uses message passing.

11.6.3 SEN Schemes

Since the design and analysis precludes the possibility of efficiently implementing optical exchange switches for a large SEN, we focus on SEN designs that use optical shuffles and electronic processors and network control. The implementation options in such a case is either a serial optical or a parallel optical shuffle. We use the term parallel to include both fully parallel and quasi-parallel data transfers. This broader definition is employed since it is not certain that fully parallel (100 bits) optical data transfers (at the board level) may be possible for a large number of processors. The actual physical size and partitioning of the PEs will dictate the level of parallelism in the message transfer. An example of such (quasi-) parallelism would be to use 25 signal lines (encoded in fewer lines or non-coded data in 25 channels) to transfer the message in four periods. Note that if the messages are sent serially on the shuffle network, it has to be operated at least 100 times or faster (for a 100 bit message) than the speed at which the PEs operate. This makes the problem of interfacing the network and the processor or processor control difficult.

We examine the serial and parallel optical shuffle implementations to note the merits and demerits in both.

11.6.4 Serial Optical Shuffle

The messages are assumed to be generated within the PE and stored in the output buffer (OB) (Figure 11.5). Each processor has access to two clocks: the first for the electronic processor and exchange circuitry and the second faster one to clock the OB, IB, and the diode array. It is assumed that both clocks are distributed optically on the chip as well as on the board. The faster clock is assumed to be almost 100 times faster than the slower one. In such a case, the network cycle time is as long as that of the processor. Effectively, the processor and network operate at the same speed to maximize throughput. Note that since the PE can be assumed to load the OB in parallel, there is no delay in moving the message within the PE. Thus, if the PEs are high performance processors designed to run on a clock of 50 MHz (100 MHz), the network clock must operate at 5 GHz (10 GHz). The complete network cycle is then about 20 ns (10 ns), although all switching within the network occurs with a delay of 200 ps (100 ps). To avoid synchronizing problems, the slower clock would be derived from the faster clock.

There are some obvious technical obstacles to the proposal. First, the PE chip has to integrate the high-speed buffers and the laser diode array. We require, at the least, one high-speed buffer, instead of separate OB and IB, which communicates with the optical network. While the laser diode array and the buffers are required to be implemented in GaAs, the rest of the circuitry, the processor and its interface to the external world, would be in Si (ECL). This is essential since ECL and bipolar have much higher levels of integration for a full-scale processor design than GaAs. Using today's integration capabilities, separate (Si and GaAs) dies can be separately optimized and integrated on a single package. While placing a single laser diode on the package is not a problem, integrating a large number of such diodes at high speeds on a single package introduces severe problems of power dissipation, thermal coupling, and electrical and optical crosstalk. The limited number of lasers that can be integrated in a package has more impact on the number of message lines (and the number of PEs) that can be put on a chip. If lower speed are used to transmit data out of the PEs, a higher degree of parallelism is possible.

While the density of lasers on the package as well as on the PWB is a problem, significant advances in laser technology, specifically in reliability, process yield, threshold current and thermal degradation, will alleviate this problem. The more serious problem, however, is one of scalability of the serial approach. As the processor technology increases its basic clock speed, for example 50 MHz to 200 MHz (in GaAs), or the messages are increased in size, the synchronous loading of messages into and out of buffers would require clock speeds of hundreds of GHzs. Such switching speeds are not possible with current or near-future technology even with GaAs. Therefore, parallel transfer of messages must be employed if good message throughput is desired. Henceforth, we will only consider the parallel transfer of messages.

11.6.5 Parallel Optical Shuffle

Because messages are to be transferred in parallel, the speed requirement of the optical logic is much less severe. Figure 11.5 shows the schematic layout for this SEN scheme. The PE array, most likely on multiple boards, may be connected to the shuffle network by guided or free space methods. For illustration, we consider fiber connections. If wavelength division multiplexing (WDM) is used to improve the parallelism in message transfer, the limit on the amount of parallelism is specified by the number laser diodes (of the required frequency) that can be integrated on-chip, the space occupied by the multiplexing and demultiplexing optics, and the required wavelength separation between adjacent channels.

Figure 11.13 shows an array of PEs connected to an array of exchange switches. In this scheme, each exchange switch contains electrooptic detectors that detect specific frequencies and polarizations of the incoming message lines. The output of the exchange modules are shuffled and fed back to the network and the PEs.

11.6.6 Alternative Partitioning of PEs for Parallel Shuffle

The scheme presented in Figure 11.13 appears to partition naturally in vertical slices, that is, a set of PEs on one board, a set of exchange and control logic on the same or another board, and the shuffle connection at the edge of the exchange and logic board. An alternate means of partitioning the PE array and the SEN might be more attractive in terms of implementation. Consider partitioning the complete architecture into horizontal slices, which are placed on separate boards. Each board is a slice of the architecture consisting of an even number of PEs and their corresponding exchange and control stages. A number of boards, depending on the total number of PEs in the architecture and the number of PEs that fit on a board, are connected by the shuffle connection. The advantage in this scheme is that it avoids routing the message wires between the PEs and the exchange switches across boards. The problem of designing the optical SEN is then redefined as one of distributing the shuffle across multiple boards. We will consider only this configuration in all future optical SEN designs.

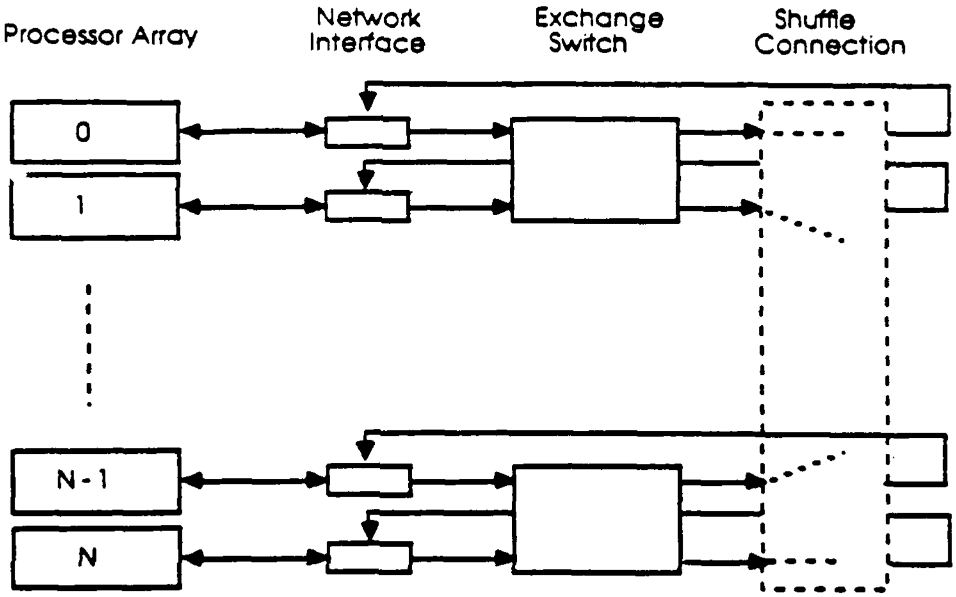


Figure 11.11 SEN and processor array

Destination 10	Mask 10	Source 10	Instruction 6	Operand 1 32	Operand 2 32

Figure 11.12 SPARO message format

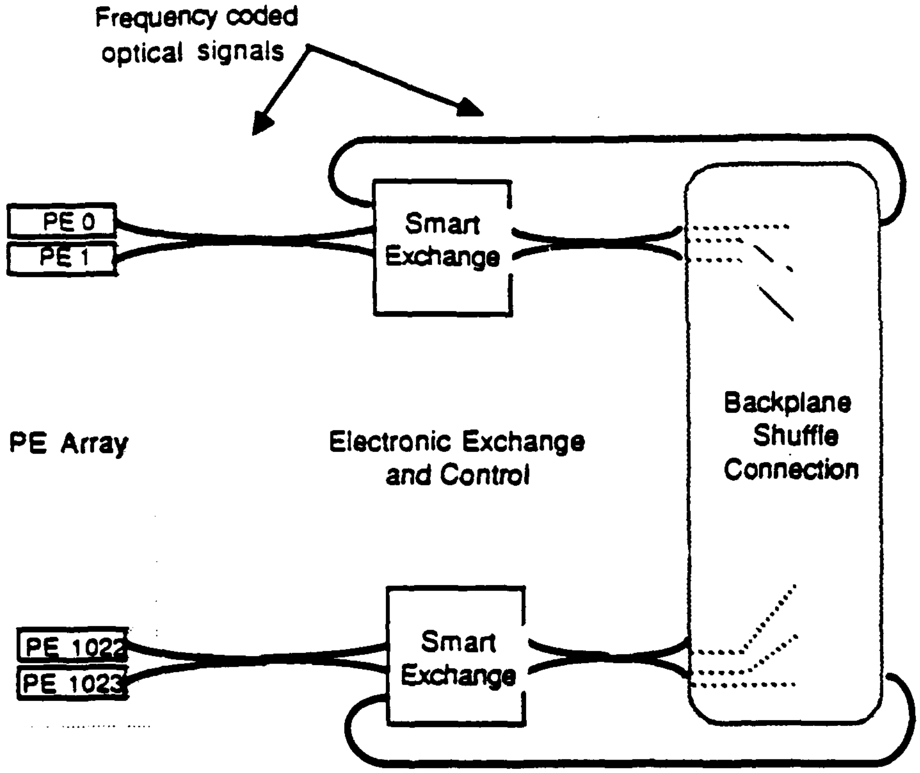


Figure 11.13 Scheme for an electronic SEN and PE array

The interboard shuffle will be specified by the number of channels on each board. This is determined by the off-board connection or wire density. The actual off-board wire density required is dictated by the number of PEs placed on a board. The number of PEs placed on each board in turn is determined by the technology used to construct the board as well as the nature of board connectors used. We examine these limits in the next section.

12. Optical and Electrical Interconnections for Fine Grained Computing

The requirements of interconnection densities depend on the interconnection network topology used. How well these density requirements can be satisfied depends on the nature of materials and devices used for implementing the interconnections. In this section we study the requirements for different topologies and how some interconnect techniques, primarily electronic, can meet them. Following this we describe in greater detail the different optical techniques that can be used to support multiboard interconnection networks.

Table 12.1 shows the possible interconnect densities for different materials and technologies. The technologies illustrated are standard board edge connector, optical fibers, Button Board electrical connectors, and multimode polyimide waveguides. Of the four technologies mentioned, only the standard edge connectors are available off-shelf. The new button board technology (using 4 mil buttons), developed by TRW and commercialized by Cinch, which connects boards on the surface is the most promising high-density electronic interconnect technology possible. At present the limit on the number of I/O points on a board that can be connected using this technology is not known, the density shown representing devices currently in development. In any case, the realizable interconnection density will be limited by wiring density within the planes to access the connectors and by electrical crosstalk between leads. The two off-board optical interconnects techniques are based on optical fibers and waveguides. Of these, the fiber optics technology is more mature. Standard fibers are usually 125 um thick. Custom fibers could be implemented with smaller diameters, several tens of microns (20 um without significant crosstalk). However, the yield in fabricating such fibers would be lower than for standard fibers because of the non-standard techniques required. Waveguides of dimensions less than 10 um and 10 um spacing provide the highest density in optical connectivity on or off-board (if alignment problems are solved). The use of waveguides, grown on polyimide substrates at Honeywell, as board level connections is still experimental. However, technology of optical waveguides is rapidly advancing. Integrated optical devices using similar technologies are now commercially available.

The different interconnect technologies can be evaluated not only on the basis of density of interconnects but also on their bandwidths. We have therefore listed the limiting speed of operations of each interconnect in Table 5.1. Complete information on the button board operation is not available yet and is being currently compiled. The advantage in the optical techniques would be their higher available bandwidth.

The exact capabilities of the various technologies will be discussed later having defined the requirements of the various networks which may be used with fine-grained systems.

Table 12.1 Interconnection capabilities of various technologies

Type of board	Density/inch	Speed	Comments
PVC/Edge connector	40	200 MHz	Standard/custom
Button Board	170	150 MHz (?)	4 mil buttons
Optical fibers	2000	Source limited	Custom
Polyimide waveguides	12000	Source limited	Experimental

12.1 Connectivity Requirements of Some Network Topologies

The motivation for implementing the perfect shuffle in optics is driven by the needs of parallel computing. These needs were derived by a detailed examination of the I/O requirements at the board level for fine-grained processing using large scale parallelism, as evidenced in commercial machines such as the Connection Machine, a SIMD computer from Thinking Machines Corporation, and NCube, an MIMD machine from NCube Computers Inc.. We find that, not surprisingly, computer architects and system designers have made major tradeoffs in the I/O design to accommodate the limitations of electrical I/O and packaging technology. Typically, multiple (as many as 512 in case of the CM) PEs are restricted to a serial I/O line, as in the NCube, or share a serial I/O line, as in the CM. Since the communication in fine-grained processing involves packet switching, the serial communication of messages between PEs imposes a severe performance restriction. Because I/O is such a severe bottleneck, the PE performance is normally degraded and state of the art technology cannot be used. The overall effect is that the throughput of the parallel computer is limited by the I/O bandwidth at the board level.

In this section we examine the connectivity requirements for different network topologies. We can show that the wiring complexity across boards, for a multiboard system, of the shuffle-exchange is no worse than that of other interconnection architectures such as the hypercube and the crossbar when parallel message transfers are considered. In fact, the board-level connection density for SENs is less severe.

Consider, for example, a large-scale parallel architecture consisting of NM PEs distributed across M boards (or clusters, in the general case) communicating via messages. Thus each board has N PEs that need to communicate to other PEs on its board as well as others. Since we are concerned with interconnection requirements across board boundaries for different interconnection architectures, we will not consider the on-board connections that can be done by board-level routing. We will examine the total I/O channels required per board for the same density of PEs on a board for different interconnections.

12.1.1 Hypercube Interboard Connectivity

Let us first examine the hypercube. Since there are a total of NM (N and M are necessarily powers of 2) PEs, the dimension of the hypercube is $D = \log_2 NM$. Thus each PE has both input and output connections to D other PEs.

If each PE communicates a B-bit data packet or message, then each PE requires DB bits for each input or output connection. To estimate how many of the D PEs are on the same board as the source PE, we have to consider the how the PEs are partitioned.

The best case, that is, when the least number of connections are required outside the board, partition occurs when each board of the N PEs form their own smaller hypercube and D is minimum or 2. In such a case, each PE on a board is connected to $\log_2 N$ PEs on the board and to only 1 PE on the second board. In the general case when there are M boards ($M >= 2$), each board contains N PEs in a hypercube and the number off-board unidirectional PE connections per PE in the best case is $2(\log_2 D - \log_2 N)$ or $2\log_2 M$.

Thus, the number channels required per PE in the board is $2B\log_2 M$.

The total number of channels for each board is $2NB\log_2 M$.

12.1.2 Crossbar Interboard Connectivity

The cross-bar connection for connecting a massively parallel PE array does not really make practical sense since all PEs are permitted to address any other PE in one cycle. However, for purposes of comparison and completeness, we will examine this interconnection topology. Each PE has to be connected all others. In the multiboard case, each PE has to be connected to all $2N(M -1)$ off-board PEs, besides the $N - 1$ on-board PEs.

Thus, the number (unidirectional) channels required per PE in the board is $2BN(M - 1)$.

The total number of channels for each board is $2BN^2(M - 1)$.

12.1.3 Shuffle-Exchange Interboard Connectivity

The computation of the I/O channels required for the shuffle-exchange is relatively simple since each PE has a fixed fanin and fanout of 1. However, in a multiboard situation the partitioning of the processors determines the number of interboard connections.

In the best case, $M=2$ and the 2N processors can be split up such that only N/2 PEs on each board require off-board connections to the other board. The rest of the PEs can be connected by the shuffle-exchange on-board. This is because the shuffle connection is bisymmetric (that is, the shuffle connections for the lower group of N PEs are mirror image of the connections of the upper group of N PEs) and half of each group, that is, N/2 PEs, is connected to half of the other group. This can be verified by examining the relation that describes the shuffle permutation $S(i)=(2i+\lfloor 2i/N) \bmod N$ of the ith PE where $0 < i <= N-1$.

The total board I/O required for M=2 is therefore 2BN/2 or BN since there are two connections (input and output) for each of the N/2 PEs connected to offboard PEs.

However, the SEN is not modular, so when M is increased beyond 2, each PE in the worst case partitioning may require a shuffle connection to a PE off-board. In such a case, each PE has I/O connections to two other PEs off-board, one for input and the other for output.

Thus, the worst case number channels required per PE in the board is 2B.

The total number of channels for each board in the worst case is 2BN. Table 12.2 summarizes the I/O channel requirements for an N PE board where M boards contain a total of NM PEs. In the same table we also show the board I/O requirements for NM = 1024, M = 8 (assuming 128 PEs per board), and B = 100. Note that the columns marked as I/O per PE or I/O per board do not reflect physical fanout but rather the required connectivity. This is because, as in the hypercube operation, the PEs do not operate in a broadcast mode but rather selectively talk to individual PEs at any one time.

Figure 12.1 shows how the board-level I/O increases as a function of the number of boards for conservative values of N and B (N = 16 and B = 32). Figure 12.2 and 12.3 show graphically the total I/O requirements as a function of the number of processors for two different sets of values of message width B and the number of boards M. In each graph we have also provided two reference lines representing the total board I/O possible in two different technologies, button boards and optical fiber interconnects, assuming that a large 18" X 15" board is used. Note that since button boards have been designed for a maximum of 2000 buttons a 8" X 6" board only, we have extrapolated that figure and assumed that 5000 buttons can be placed on the larger board. In case of optical fibers, we have assumed that they are used only on one edge of the board, and not on the complete periphery like the buttons on the button board. From size and spacing considerations, 36,000 optical fibers can be fitted on the 18" side of the board. The figure is even better (216,000) if waveguide connections can be used on the edge of the board. Thus for a large number of PEs, optical interconnects appear to hold more promise than available electronic techniques.

For reference, we have provided three (connected) points in Figure 12.3 that reflect the current and projected board I/O requirements of one 8K card rack (containing 16 cards with 512 PEs each) of the 64K CM. The lowest point (768) represents the current offboard I/O required in each board, where every 16 PEs share one serial link. The second point represents the 4K offboard connections required if each PE were allowed its own serial link. The third point represents 184K connections required if each PE

Table 12.2 Crossovers for several interconnection networks

Interconnection	I/O	I/O	Normalized I/O	Board I/O
Topology	per PE	per Board	per Board	(N,M,B=128,8,100)
Hypercube	$2B \log_2 M$	$2NB \log_2 M$	$\log_2 M$	76.8 K
Crossbar	$2BN(M - 1)$	$2BN^2(M - 1)$	$N(M - 1)$	22.4 M
SEN (M = 2)	B	BN	-	-
(M > 2, worst)	2B	2BN	1	25 K

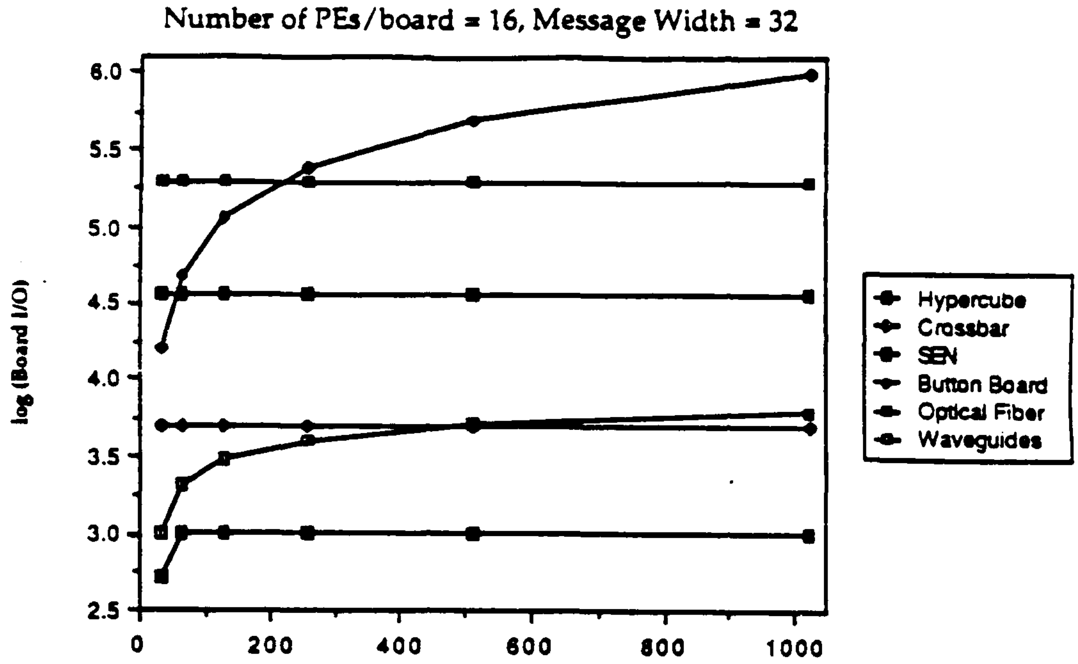


Figure 12.1 Board level I/O as a function of the number of boards for B=32, N=16

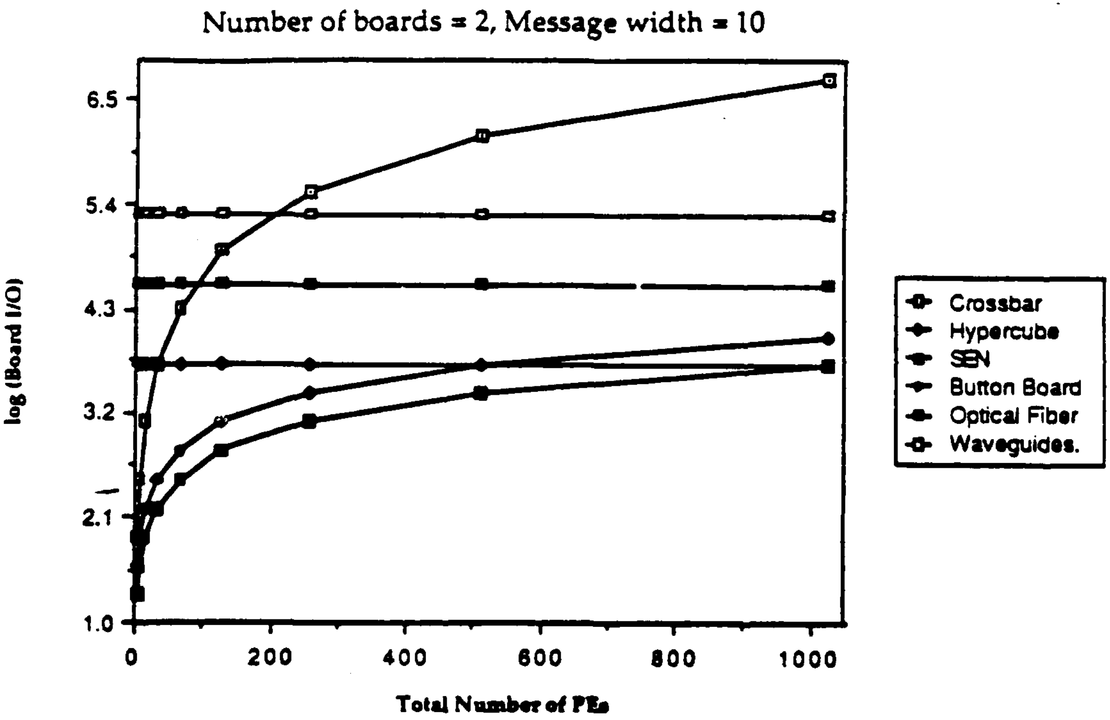


Figure 12.2 Board level I/O as a function of the number of PEs per board for B=10, M=2

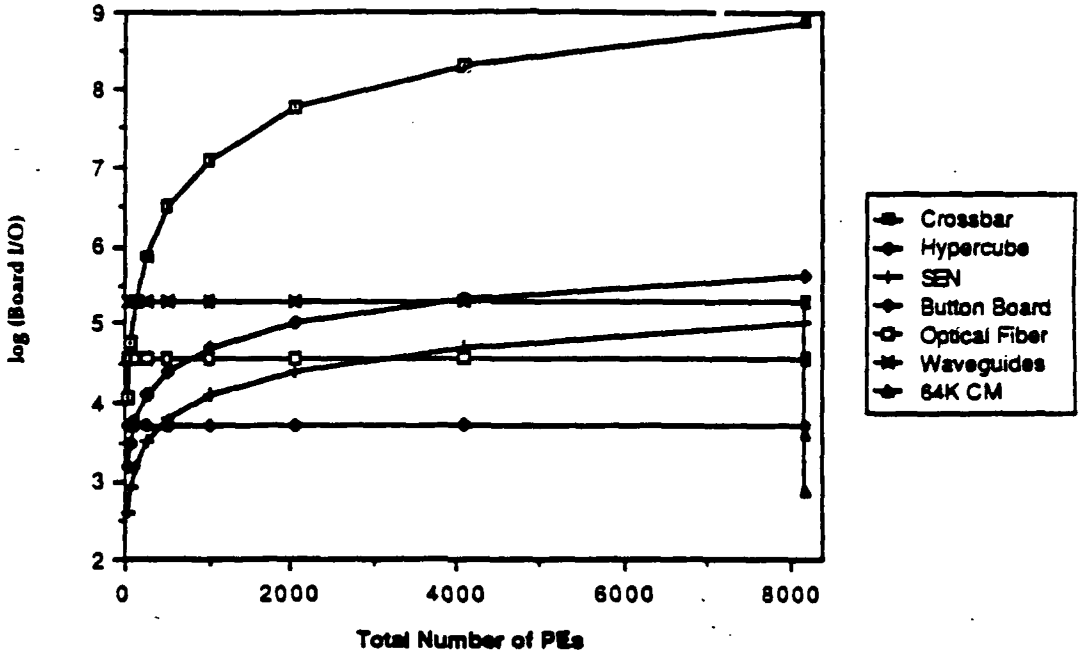


Figure 12.3 Board level I/O as a function of the number of PEs on a board for B=100, M=16.

were allowed 46-bit (for 4 bytes of data in a packet) parallel messages.

When comparing the three networks, we find that the crossbar, because of the quadratic increase in the number of I/O channels is beyond practical consideration for large networks. The SEN fares better than the hypercube-an optical fiber approach can support less than 1000 PEs in a hypercube as opposed to 3000 PEs in a SEN. The higher I/O density in the hypercube is due to the larger fanout, by a factor of $\log_2 M$ over the SEN, of each PE. Further, in case of a SEN, these limits on the number of PEs per board scale only with the total number of PEs, unlike in the hypercube where the connectivity requirement of the board increases with the number of PEs as well as the number of boards. We note, in fairness, that the limited connectivity of the SEN, implies handling a smaller load, i.e., usually around 25%. Larger loads would slow down message deliveries since more conflicts would occur. For significantly higher message traffic, replicated networks are recommended.

Having examined the board I/O densities of different networks from a topological and computational perspective, and having previously concluded that electrical interconnection networks are inadequate in meeting the requirements, we examine how different guided and free-space optical interconnect technologies can meet those requirements, and which appears most promising.

12.2 Board-Level Interconnect Technologies in Optics

To address the demand of high I/O density at the board level that we established in the previous section, we embarked on analyzing different optical interconnect technologies as possible candidates. The key optical technologies that we investigated are fibers, polymer waveguides, volume holograms, planar holograms and microoptics and bulk optics

To assess the relative capabilities of these technologies with the existing electrical means, we have also examined two electrical board-level interconnect approaches considered in the previous section. These are high-density conventional connectors and TRW's button boards. Before discussing the merits and demerits of the different optical approaches, we have first listed a number of issues that were used as criteria for comparing and assessing the optical approaches against the existing electrical interconnect technologies.

12.2.1 Issues in using optical interconnection

The following issues are of concern when optical interconnections are used. These considerations are important if optics is to provide a competitive edge over electronic board interconnect technologies. An evaluation of these issues provide the fundamental physical and technological limits in using optical interconnects for boards.

i) Power: Power considerations are necessary primarily for transmitters, detectors and receivers; the power consumed by the interconnect circuitry directly determines, at least in part, the density of optical interconnects on the board. The power budget is a function of the optical losses in the system and the power required by the drive circuitry for the sources (lasers and /or modulators) and the receivers.

ii) Size and volume of optical components: This refers to the size of optic devices (lenslets, mirrors, receivers, etc.) and the volume of the hologram, if holograms are used for connecting signals at the edges of boards. The size of the optical subsystem is important since it determines the optical path length and therefore the delay in the system. In addition, in situations in which individual optical elements determine the path of a given channel or set of channels, the size of the components determines the intra board connection density attainable.

iii) Density of receivers and transmitters: The physical size and density, in case of integrated devices, determines the density of I/O connections possible. The fabrication technology also determines what densities are practically achievable. Thermal considerations and optical and electrical crosstalk also limit the density of connections.

iv) Speed and Bandwidth: The speed and bandwidth of the receiver and transmitter determine the I/O bandwidth of the system. The usabel data rate of the optical interconnection is also affected by the dispersion of the interconnection medium, where paths other than those defined in free space are used.

v) Crosstalk: Crosstalk is important in determining signal-to-noise ratios and bit error rates in optical communications. Because of lower efficiencies in transmitting information, crosstalk is much more severe in degrading noise margins in optics than in electronics.

vi) Tolerance: The tolerance in the physical dimensions of devices and components is crucial if boards can be pulled out and reinserted into the racks. Since optical interconnects for parallel systems offer the possibility of near-term incorporation into machines, their use should not require abandoning establishes concepts for multiboard systems. The tolerance of change in the wavelength of the signal as well as the temperature of the source is also important when ensuring correct connectivity.

vi) Reliability: The lifetime of all devices used is especially important when competing with mature electronic technologies. The large number of independent channels for a massively parallel system communicating with parallel bits places stringent requirements on the performance of individual devices.

vii) Cost: For practical considerations, the cost of interconnecting processors distributed across two or more boards must be close to that of electronics unless the density of optical interconnects far exceeds that possible in electronics.

12.2.2 Optical interconnect comparisons

To implement the perfect shuffle connection between PE boards optically, a variety of approaches are possible. These include volume holograms, planar holograms, waveguides, optical fibers, and microoptics. Each of the interconnect formats will be considered in turn together with their relative advantages.

Volume holograms would appear to offer the possibility of board to board connection with high density. To most effectively use the hologram, a vertical connectivity would be employed. An array of vertically emitting sources would be mounted on the lower surface of the upper board. Immediately below it would be a volume hologram, which would serve firstly to collimate the output from each source, and to direct the signals from the sources associated with a particular exchange switch output to the relevant set of detectors. Thus the hologram is divided into a set of facets, one for each set of sources associated with an exchange switch output, and within each of these facets an array of smaller structures serving to collimate the output of each source.

Several design requirements conflict. In order to achieve high resolution of the image plane, a large hologram is required. High density in the hologram plane requires small holograms. High density in the source plane requires that the hologram plane be close, to prevent the diverging output from the source (laser or modulator) reaching the hologram associated with the adjacent channel. A study of literature evaluating volume holograms for inter-board distribution indicates that for geometries likely to be encountered in intra-board connection, densities of 100 per square cm may be attainable.

Configurations involving multiboard connections with holograms located between boards are cumbersome, and subject to even lower densities of connectivity. Even in the two-board case, the hologram must be located accurately with respect to the source array, both positionally and with respect to angle. Although the deflection associated with each hologram facet is locally space invariant, each smaller facet also serves the purpose of collimating a source, behaving in a manner similar to a lens. Thus significant spatial misalignment of the hologram will result in angular misalignment of the beam. Smaller misalignments will result in unacceptable crosstalk.

Holograms used in this application impose stringent requirements on source wavelength. If the hologram is fabricated to operate at a given wavelength, the source wavelength used must be sufficiently close to this value. Acceptable tolerances are approximately 1nm. This implies a stabilization of the source laser temperature to approximately 1 degree for a simple Fabry-Perot laser and perhaps 20 degrees for a DFB laser. The DFB lasers are more complicated, and therefore more expensive to manufacture. LEDS are obviously unsuitable. Fabrication of vertically emitting lasers with such close tolerance has neither been demonstrated nor reported in the literature. In addition, control of individual laser wavelength within an array of uncoupled lasers is likely to be difficult, and require space-consuming circuits.

Use of matched pairs of holograms can offset misalignment due to temperature changes. However, more fundamental problems exist. As will be shown later, reliability considerations suggest that individual lasers will not be used for each channel. Thus approaches involving fanning out of a remote laser to a number of modulators are likely to be employed. While the fanout is easy to implement using holograms, the spatial light modulators employed for such modulation have to date demonstrated inadequate speed and extinction ratios, or have required high drive voltages. Effron [62] presents a comparison of available spatial light modulator types.

Having eliminated holograms between boards on the grounds of scalability and tolerance, one is tempted to consider the use of holograms in the backplane. Here, edge emitting structures are permissible. However, the achievable density is now only 10 per linear cm, since only one of the two orthogonal dimensions of the hologram is used.

12.2.3 Fiber optics

An extremely simple means of providing the optical shuffle within the interconnect medium between boards is by the use of optical fibers. Here an array of optical outputs are connected to a ribbon of optical fibers. Each ribbon corresponds to the outputs from each channel of the exchange switch, assuming no multiplexing. Thus for a total of 128 PEs on two boards, 32 ribbons would be required. High density fiber connectors are available commercially [63] with 18 fibers per connector, and the techniques used in their construction are readily scalable. However the approach is inelegant, and requires increasingly cumbersome assembly as the number of PEs increases. Acceptable bend radii are typically several centimetres, hardly attractive for the systems under consideration.

The technology for fiber optics is however one of the most mature of those considered here. Fibers are low in cost, and connector technology is mature for low densities of connections, and has been demonstrated for high densities. For higher densities than are implied by the 125 um outside diameter of most fibers, modified manufacturing technologies would be required, resulting in a higher cost. Other technologies may therefore offer lower costs per channel. A state of the art demonstration might have a density of 80 per cm. Etching of existing fibers might yield a density of 500 per cm, while alternative fibers could be developed with very high relative refractive index differences to yield a maximum density of 10^3 per cm. These limitations for our high density application reflect the development of fiber optics for long distance point-to-point links, where space is rarely a consideration.

12.2.4 Polymer waveguides

These waveguides, developed earlier at Honeywell, have the advantage over fiber waveguides in that the processing involves planar techniques, and the waveguides could be fabricated directly on printed circuit boards. Waveguides have been demonstrated with dimensions of approximately 10 mm in width and 4 mm in thickness, and on a range of materials, including other polymers, glass, silicon, and Aluminum. These are multimode. Adjustments to the guide dimensions are easily incorporated to increase the waveguide pitch. With such adjustments, a theoretical density of the order of 1,000/cm is predicted. A state of the art demonstration would today have a density of 250 per linear cm.

The crosstalk between parallel waveguides is negligibly small for waveguides with spacings comparable to the waveguide widths involved, and with the high relative refractive index differences involved when the adjacent guides are surrounded by air. Of more concern is the crosstalk associated with the intersecting waveguide junctions which will be required in the perfect shuffle implementation with waveguides on one layer. For N PEs on each of two boards, each with 100 channels, depending on the particular configuration used, on the order of 100N crossovers will be required in the worst case. (This is proved later. Again using a value of 128 for N, and assuming worst case number of crossovers, we find that to preserve a crosstalk of -20dB at the output, each crossover must on average have a crosstalk of -60dB. To date, crosstalk figures have been -51dB per intersection. This suggests that multlayer waveguides should be employed. It would not be necessary to fabricate all layers simultaneously, merely to assemble the complete waveguide assembly from several single layers. This problem becomes more critical for systems using more than 1000 PEs, as would be the case for a massively parallel system.

Figure 12.4 shows the shuffle connection across two boards for a network of size 8. Without making any attempt to optimize the connection layout, it is seen that two of the crossovers are associated with outgoing lines crossing lines associated with each board, while the two remaining crossovers are associated with lines between boards. Light in these intersecting junctions propagates in opposite directions. An important distinction should be made between the two types of crosstalk. In coupler terminology, a 4-port coupler illustrated in Figure 12.5 connects inputs 1 and 2 to outputs 3 and 4. In our case the ratio of the powers of the signals, P1 to P4 and P2 to P3 should be unity, while all other coefficients should be zero. Crosstalk, defined specifically as P3/P4, may arise from non-optimal designs, while a finite directivity P3/P2 may arise from a different set of imperfections. In many couplers the directivity is significantly better than the crosstalk. While sufficiently accurate data are not available for polymer waveguides, surveys of results obtained from other waveguide technologies such as high-index silica, ion-exchanged glass, or $LiNbO_3$ indicate that values of 40 or 50dB may reasonably be expected.

Thus, the implementation of the perfect shuffle using polymer waveguides is possible, but various forms of crosstalk will limit the ultimate scalability. It should be pointed out that "engineering" techniques may result in lowering the total number of crossovers, but that no rule exists for determining the optimal layout. The situation is analogous to the CAD layout of printed circuit boards, and therefore layout and routing algorithms from the CAD world could be profitably adapted to reduce the number of total crossovers. A simple solution is currently under consideration for the case of N PEs on each of 2 boards.

Figure 12.6 shows micrographs of polyimide waveguides exhibiting certain features, including 45 degree and ninety degree waveguide bends. The sizes of the waveguides illustrated are approximately 25 microns by 10 microns (height). These waveguides were not developed under this program, rather on a DARPA/NOSC program. The data are merely included here to indicate the feasibility of the solution, and to enable us to justify our assumption that the required interconnection medium will exist.

12.2.5 Planar holograms

An alternative approach involves the use of planar holograms. These are equivalent to modified planar waveguides, either single mode or multimode. Predicted densities are 10^2 per square cm for the multimode planar holograms, and 10^4 per cm for the single mode version. The geometry here differs from that of the volume holograms since planar holograms control the propagation of light

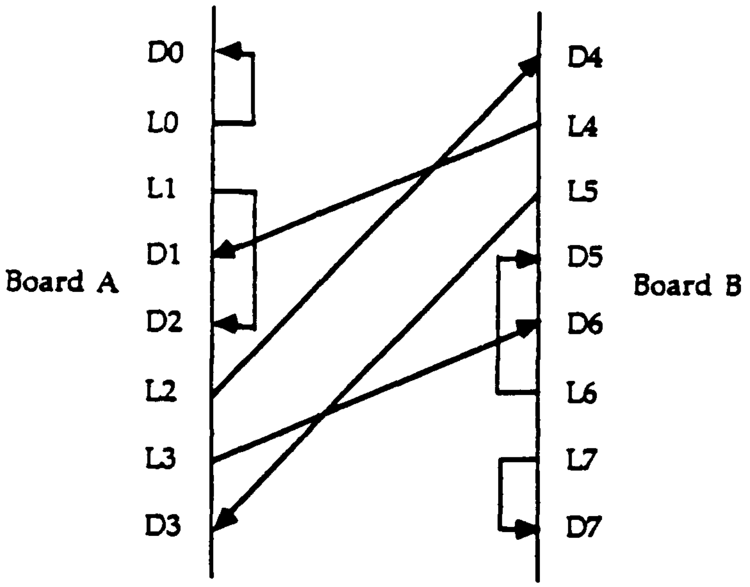
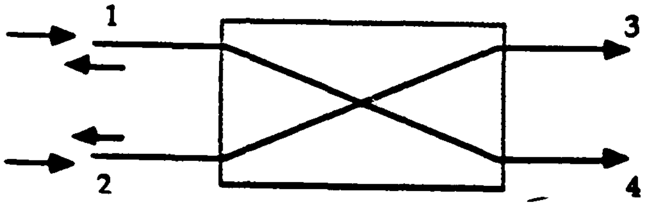


Figure 12.4 Waveguide shuffle connectivity across two PWBs.



Ideal: P4/P1 = P3/P2 = 1

Crosstalk: P3/P4 for input into 1
P4/P3 for input into 2

Directivity: P2/P4 for input into 1
P1/P3 for input into 2

Figure 12.5 Crosstalk across two intersecting waveguides

from one edge of the hologram to the other. Crosstalk is predicted to be approximately -20dB at these densities. Thus, the planar holograms offer an advantage if the number of crossovers is greater than 100.

The problems associated with mechanical alignment of the holograms are not as severe as for the volume holograms since planar alignment techniques may be employed in this two-dimensional case. However, the concerns of source wavelength and its stability are still relevant.

Although an N-point to N-point distribution has not been demonstrated with this technology, reported results of a fanout demonstration indicate that a state-of-the-art demonstration might involve 10 connections. The state of the art demonstration appears to be about three orders of magnitude less than the theoretical limit. Further evaluation of this technology is necessary. Issues such as the scaling of crosstalk with the number of point-to-point connections have yet to be resolved. At present this technology is not sufficiently mature to enable a cost to be associated with it.

12.2.6 Bulk Optics and Microoptics

Lohman has reported a perfect shuffle mapping in optics using bulk components. Provided all sources and all destination processors lie neatly in a line on opposite sides of the lens, the mapping is easily implemented. Folded versions have been reported which increase further the attainable interconnection density. For parallel operation, anamorphic optics would be required to ensure that a number of parallel channels were subjected to the same mappings.

For PEs distributed across a number of boards, the mapping becomes insufficiently regular for a single lens to perform the routing. Instead, a single lens would be associated with the output from each processor. Use of such a large number of lenses, and the accurate tolerances involved, render this approach unattractive. In addition, the arguments relating to speed of spatial light modulators further reinforce our decision not to use this technology. Since bulk optical implementations of the perfect shuffle have been demonstrated, one would expect microoptic versions to be possible.

Using miniature prisms and gradient index lenses ,the required mapping could be performed. Issues not yet resolved concern the effect of aberrations present in real lenses, and the input/output density attainable with this format. The negative comments concerning discrete bulk optical components and the labour involved in their assembly and alignment also apply here. Microoptics does however offer significant advantages for use in connectors between boards and backplane when used with polymer waveguides, for example.

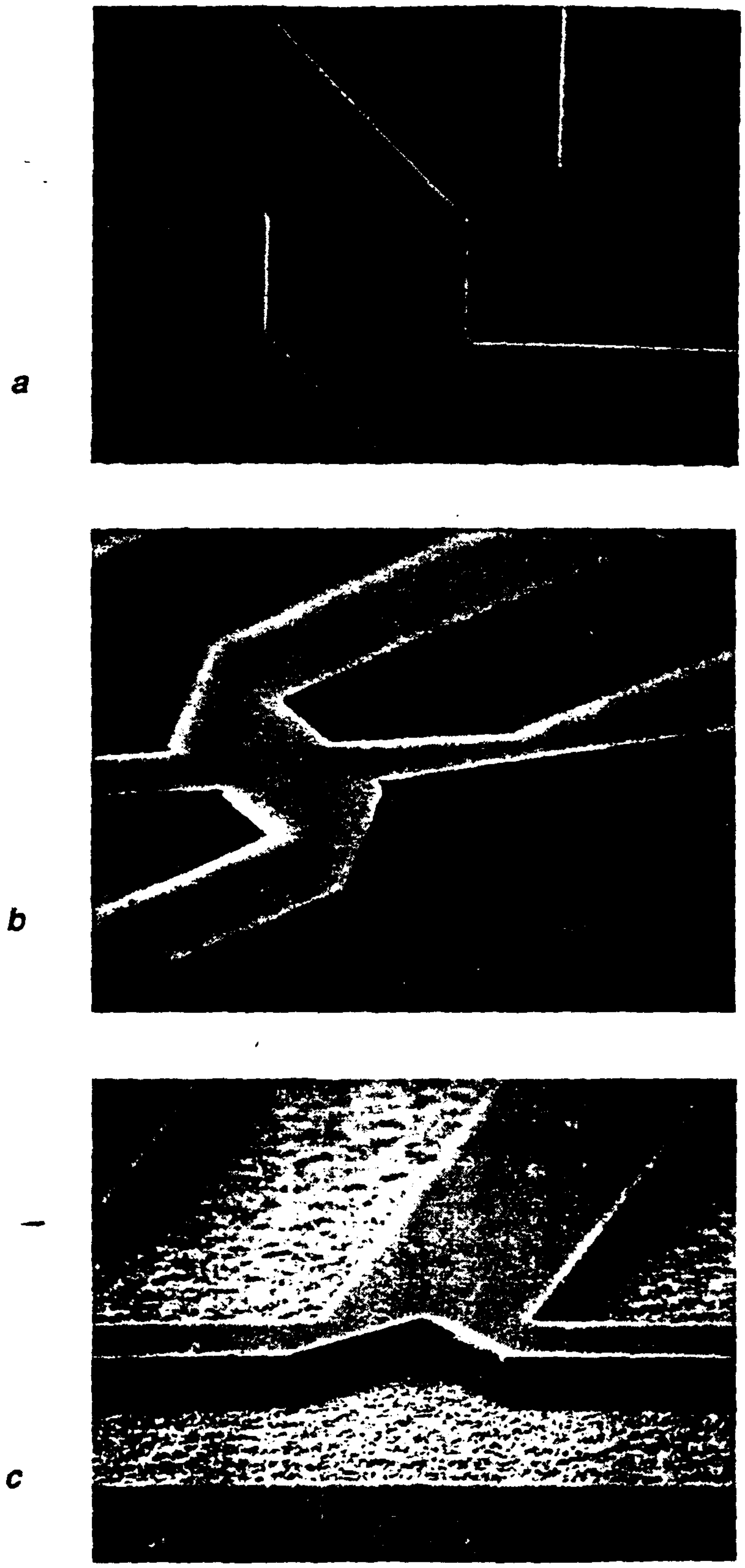


Figure 12.6 Polyimide waveguide components: (a) two 45° bends and one 90° bend, (b) two 45° bends and a 90° intersection, and (c) a 1x2 splitter

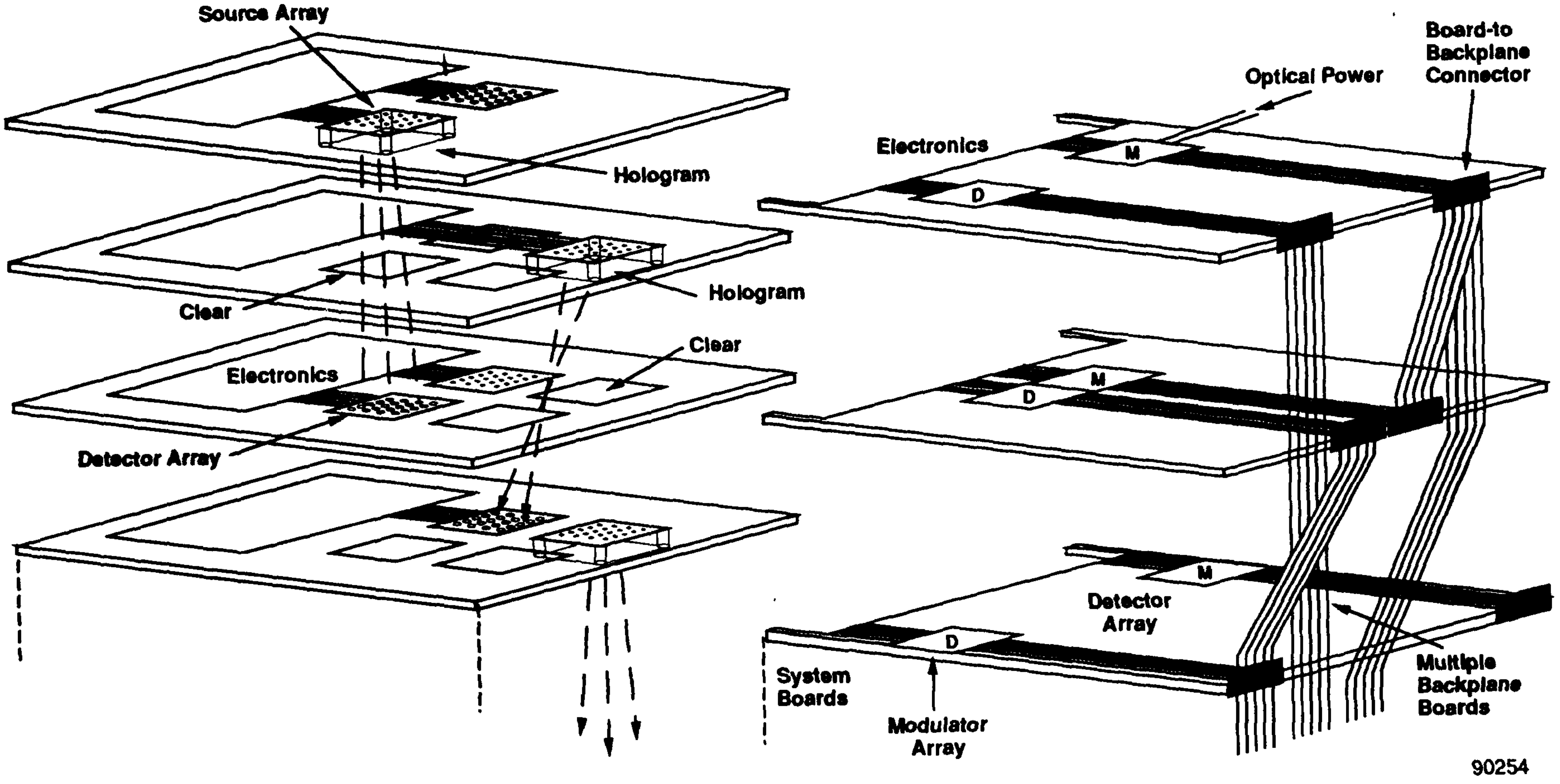


Figure 12-7. High Density, Free-Space and Guided Wave Interconnections for Fine-Grained, Parallel, Multi-Board Systems

12.3 Summary of optical backplane approaches

On grounds of loss, bandwidth, board fabrication compatibility and ultimate low-cost per channel, we infer that the most favored approach is that of the polymer waveguides. The ultimate scalability will depend upon the performance of individual crossovers and other components. Most high-quality sources under consideration are edge-emitters, while most detectors are in a planar form. The polymer waveguide approach is compatible with both, given that vertical reflecting surfaces have been reported in the literature [64].

Figure 12.7 shows the likely format of both free-space and guide-wave implementations of the optical hybrid interconnection scheme.

Planar holograms deserve further investigation as the technology matures. Their use for the shuffle exchange, in conjunction with polymer waveguides, to transfer signals across relatively large distances may extend the scalability. However, this technology is significantly further behind in development that competing approaches.

In order to demonstrate a connection scheme using such waveguides, a connector would be required between the backplane and each board. Expanded beam connectors appear to be suitable candidates, but further work would be required to design a suitable connector.

12.4 Sources and detectors for the optical shuffle connection

The high input/output densities described here imply the integration of many sources and detectors on one chip, rather than the assembly of discrete components. We consider first the source issue. Choices for the transmitter section of the network include LEDs, lasers, and optical modulators driven by external sources. Issues involved in choosing the best approach include power dissipation of the source and drive circuitry, reliability, sensitivity to change in operating environment, yield in fabrication, and cost. We consider each possible solution in turn to determine the most appropriate for our needs. Our comparison is centered on short-wavelength (say 780-830nm) sources, since the distances involved are such that modal dispersion will not become a problem unless data rates per channel exceed 10GBits/sec. Sources and detectors at the shorter wavelengths generally have lower cost and higher performance. The bandwidth of the polyimide waveguide may be expected to be 100GHz.cm, so that increasing throughput would best be accomplished by increasing the message width. This highlights another potential advantage of non-serialized interconnection approaches, other attractive features including the avoidance of latency in multiplexing which would otherwise be incurred.

LEDs typically have low output power, and such power as is emitted is distributed across a large area, resulting in little transfer of power into a coupled waveguide. Also, efficiencies of only one or two percent result in large amounts of power being required to drive the device for a given optical output level. The large amount of drive power therefore required to obtain bit error rates of 10^{-15} required in such systems would make this approach unattractive. In addition, their speed is low, perhaps 100MHz.

Typical semiconductor lasers operate at currents of the order of a few tens of milliamps drive current. Typical threshold currents are a perhaps 10 milliamps, while ultra low threshold devices are currently being developed, having attained several milliamps to date. Modulation of diode lasers requires switching perhaps 10 milliamps for each laser in the transmitter array. Significant efforts have been expended in the enhancement of laser reliability. However, at the short wavelengths, dark line defects generally limit the device lifetime. Reliability data of Fabry-Perot AlGaAs lasers indicate that the failure rate of these diode lasers increases with operating time. We have extrapolated the available data to describe the reliability of the 128,000 sources likely to be employed in a system with one source per channel. It was found that even at room temperature, a hard failure of one link would occur within hours, on average. It was assumed that the time and ensemble average failure rates were identical.

Given such a large number of sources, incorporation of full redundancy is infeasible. On grounds of reliability and ease of incorporation of redundancy, an approach involving a small number of discrete lasers driving a number of arrays of modulators is preferable to one involving a large, high-density array of uncoupled lasers. Issues involved in such a design are the fanout between lasers and modulators, the available power of the lasers, the effects of high power densities on the input to the waveguide fanout, loss in the fanout and the modulators, and the optical and electrical characteristics of the modulators. In either arrangement, careful design would be required to reduce electrical crosstalk and therefore optical crosstalk. Some circuitry is required to drive the modulators -heat dissipation in such a circuit and the size of the circuit will dictate the usable density of the transmitters.

Waveguide modulators are voltage controlled devices, the current required for switching being small, and being determined by the switching speed and the device capacitance. For example, suppose a given laser requires 30mA operating current, with a bandgap corresponding to 1.5 Volts. Typical CW power dissipations will be approximately 40mA. Since the shuffle exchange network works particularly well with low network traffic, we assume a duty cycle of 10%. Thus each laser must dissipate 2mW on average, but must still be able to tolerate 40mW. The drive electronics must be able to supply and switch 30mA at whatever rate is required by the system. We assume a value of 500MHz.

To compare the power dissipation and drive power requirements of a laser with those of a modulator, we consider a modulator with a drive voltage of 5 volts, and a capacitance of 0.5pF. Each charge or discharge of the device requires a charge of 2.5pC, or supplying an energy of 12.5pJ. At a target data rate of 1Gbit/sec with a NRZ format, the modulation frequency would be 500 MHz. Thus the power to be supplied must be 6mW. This has considered the modulator to be a lumped electrode device. At higher frequencies, traveling wave configurations would be required, necessitating termination of the device in its characteristic impedance or its complex conjugate. For an impedance of 25 ohms, the power dissipated in each terminating resistor would be 1Watt. The excessive total power dissipation of terminated structures suggests that lumped and unterminated structures should be used provided severe penalties are not incurred in the drive requirements.

For the modulator, the driver must be capable of supplying an average current of at least 1mA. In addition, the driver must be able to provide sufficient current to overcome the loss of power in the series resistance. For a series resistance of 10 Ohms, this would be approximately 10mW. The laser driver however requires that perhaps 10mA be supplied in the on state, with less in the off state. Let us assume that 2mA corresponds to threshold, then the average power dissipation for a 50% duty cycle would be approximately 10mW.

Modulators then do not necessarily have any advantage in terms of drive power compared to lasers, and indeed as the frequency increases show a pronounced disadvantage in cases where the drive requirements are limited by the modulator capacitance.

Disadvantages of the modulator approach are the loss sustained by light propagating through the modulator, and the losses associated with interfacing the modulator into the optical network. In addition, two extra interfaces are introduced compared to the laser, thus an extra critical alignment step is introduced.

To realize the potential benefits of modulators over lasers, and to minimize the overhead of packaging, it is proposed that a remotely mounted laser be used to provide optical power to a number of modulators. The optimum fan-out of the source will be determined by considering the loss for the complete system, and the properties of the modulators and receivers. To enable the system to be inserted into a system,it is envisaged that the modulator array and fan-out chip would be assembled into a hybrid package.

For logic-compatible modulators in III-V materials, devices of a few millimetres in length are required. At speeds of 1 GHz or less, traveling wave configurations are not required, and the electronics drives a mainly capacitive load. The modulator design can therefore be relatively simple in comparison to modulators for telecommunication applications, since frequency chirp is not an issue in the low-dispersion short length waveguide connections.

Similarly, in the receiver array, the design of the amplifier circuits rather than the photodetector will be the limiting factor. For example in reference [80], a monolithic receiver circuit is described with a cell dimension of 75 um x 175 um, although the detector only occupies an area of 10 um x 10 um. Some reconfiguration of the layout of this circuit would enable high transverse packing densities to be achieved, up to 10um pitch. The crosstalk which would result has not been considered. Approximately, 75 mW were dissipated in this circuit. Allowing a more conservative packing density of 200 per cm arising from a receiver pitch of 50 um, 15W would be dissipated per linear cm, or approximately 160W per square cm. This is at the limit of what can be tolerated with the most advanced heat sinking technology. In any case, the lifetime of the devices at the resulting elevated temperatures is questionable. Tradeoffs between receiver complexity and performance may be performed. Such an optimization would include the optical performance of the backplane connection and the performance of the sources or modulators.

Our initial analysis of state of the art detectors and receivers, such as the one above, reveals three problem areas that must addressed before board-level optical interconnects can become competitive with electronic connections. These are:

i) size of the receiver circuit,

ii) crosstalk - both optical and electrical, and

iii) power budget for the receiver

Each of the size, crosstalk, and power is currently too large for implementing high-density and high bandwidth board-level interconnects. This is again a consequence of the bulk of developmental effort being focussed of telecommunications type receivers, where space is not at a premium, and large amounts of electrical power may be consumed to compensate (within certain limits) for low received optical powers.

In summary, a demonstration of a high-density interconnection cannot be performed unless high-density transmitters and receivers can be demonstrated with the required performance. We note that the problem of designing such transmitters and receivers is orthogonal to the actual approach used for accomplishing the backplane connection in optics, and that any optical approach employed must provide adequate solutions to this problem.

12.5 Crossovers in Waveguide Interconnection

In considering optical interconnections for different network topologies, two different approaches to realize the connectivity between boards of PEs were considered: a free-space connection approach using either holograms or bulk optics, and a guided approach using waveguides in the third level interconnect. While the free-space approach is limited primarily by density of resolvable points on the board, the guided approach is limited by the amount of crosstalk that can be tolerated by the waveguide. The crosstalk results from an unavoidable number of crossovers a waveguide channel experiences in realizing the planar layout for the network connection. Here we will examine the limitations in designing a guided optical approach for the SEN and hypercube interconnection networks using polymer waveguides.

It can be seen easily that the number of crossovers in laying out any network topology varies significantly with the number of boards and the total number of PEs. Furthermore, as experience form CAD layout and routing has shown, there are many ways of minimizing the number of crossovers, but the optimal layout problem is provably NP-hard. We will therefore focus on deriving the worst case number of crossovers that can arise for an arbitrary number of PEs, N, distributed over an M boards, communicating by messages of width B bits.

12.5.1 Shuffle Connection

The connectivity requirements of the shuffle connection is given by the mapping S,

> $S(i) = (2i + \lfloor 2i/N \rfloor) \bmod N$ where $\lfloor 2i/N \rfloor$ represent the lower floor of the value $2i/N$, and i $(0 < i <= N-1)$ is the index of the PE.

While there may be some heuristic techniques in partitioning the N PEs into P (=N/M) PEs in each of the M boards, we will not focus on that research problem. We will assume that contiguous pairs of PEs are placed on each board. Thus, the first board will contain the PEs 0 to (P-1), the second will contain P to (2P-1), and so on. Note that an even number of PEs is required to be placed on a board (even in case of a different partitioning) since the exchange switch required for every pair of PEs must be resident on the board.

To determine the crossovers we have to topologically represent the board on which the third level interconnects are placed. Figure 12.8 represents the placement of PEs from the M boards on 1 plane. Each row corresponds to the PE I/O connections in each board. There

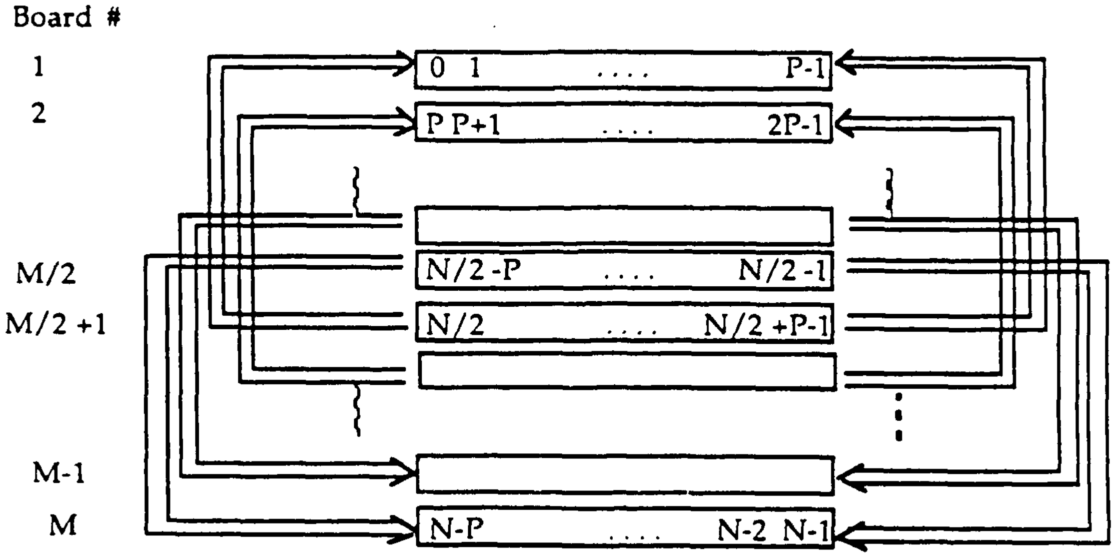


Figure 12.8 Topological layout for the Shuffle connection

are thus M rows containing P PEs each. The numbering of the PEs is shown within the row: row 1 contains PEs 0 through P-1, . ., row M/2 contains PEs N/2 -P through N/2 -1, row M/2 +1 contains PEs N/2 through N/2 +P-1,. . ., and row M contains PEs N-P through N-1. The connection of the output signals of any PE is determined by its shuffle connection.

Using Figure 12.8 as a guide, we examine the nature of connections between different boards. Note that the output of all PEs on the first board are connected to PEs in the second board. As we go down the row to the board labeled M/2, we will notice that the output connections are to boards that are further away. Thus, PEs of board 1 connect to PEs of board 2, PEs of board 2 connect PEs of board 3 and 4, . . , and PEs of the board M/2 connect to PEs of the Mth (the last) board.

Using the symmetry of the shuffle connections we find that the output connections of the lower half boards are mirror image of the connections of the upper half boards. Thus, the output of PEs of the Mth board are connected to the M-1 th board, and the PEs of the M/2 +1 th board are connected to the first board.

The distance between connecting boards, as can be shown using the shuffle permutation, depends on the relative values of P and N. The number of crossovers of any line or channel is determined by the nature of the planar layout. To decrease the number of crossovers, we will use both sides of each row for routing the connecting channels as shown in Figure 12.8. To determine the number of crossovers, note that any outgoing channel from a PE (interchangeably, any incoming channel to a PE) has to possibly cross the outgoing channels of the PEs in the same row as the source, cross channels of other connections outside the rows of PEs (left or right), and the again the channels in the row of the destination PE. The worst case occurs, evident from Figure 12.8, when the source or destination PE is in row M/2 or M/2 +1. This is because PEs from these boards have to be connected to PEs furthest from the board.

To calculate the worst case number of crossovers, we consider the individual components:

Worst case number of crossovers in the source row = PB/2

Worst case number of crossovers external to the rows = (M/2 -1)PB/2 (in the lower half) + (M/2 -1)PB/2 (in the upper half) = (M/2 -1)PB

Worst case number of crossovers in the destination row = PB/2

Thus, the total number of crossovers S_{SEN} = NB/2

12.5.2 Hypercube Connection

We will use a similar approach to derive the worst case crossovers for the hypercube connection. However, we note the following differences. Each board of the hypercube connected system is its own hypercube of dimension $\log_2 P$. A PE on each board will be connected to $\log_2 M$ PEs, each on a different board.

Using the same components of crossovers as before, we determine the worst case by considering the maximum number of crossovers internal and external to a row. We obtain the following bounds.

Worst case internal crossovers in a row in the source board $= (P/2)(2\log_2 M)B$ (the factor 2 appears since unidirectional channels are assumed)

Worst case number of crossovers external to the rows = $(M-1)P/2(2\log_2 M)B$

Worst case internal crossovers in a row in the destination board = $(P/2)(2\log_2 M)B$

Thus, the total number of crossovers S_{HYP} _ $\log_2 M(N+P)B$

Figure 12.9 shows the total number of crossovers as a function of N for both the SEN and the hypercube.

Thus, the number of the crossovers for the SEN and hypercube are $O(N)$ and $O(N\log_2 M)$, respectively.

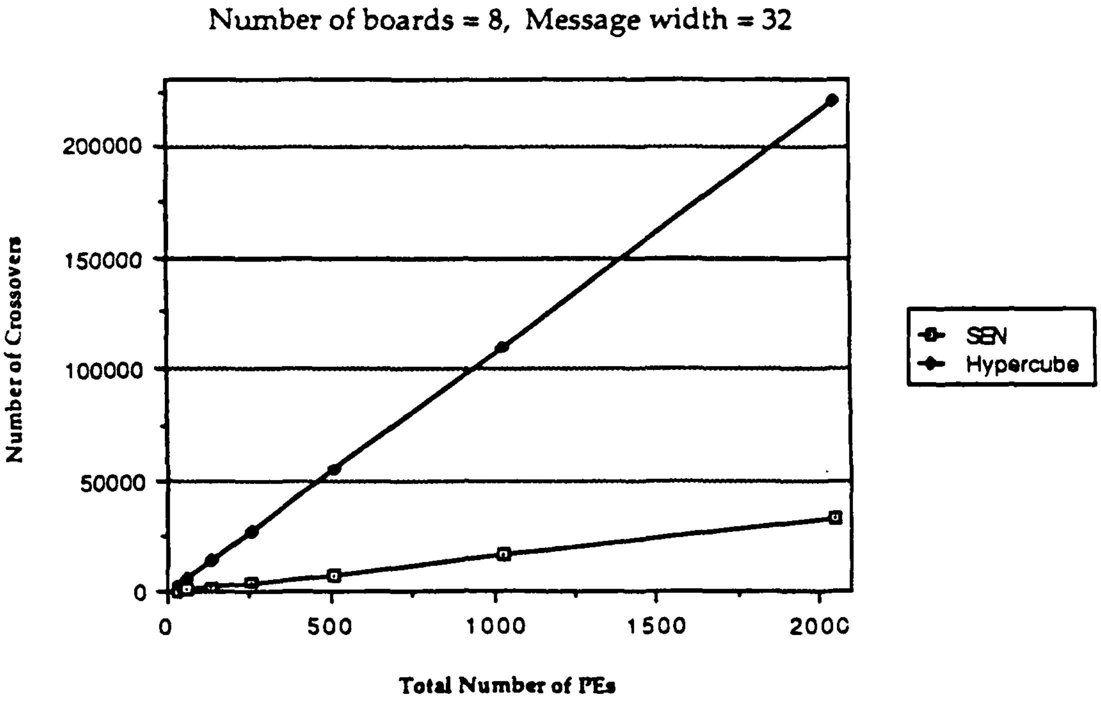


Figure 12.9 Worst case number of crossovers for third level planar interconnects (M=8, B=32)

13. Hybrid Optoelectronic Implementations of Multiboard Shuffle Connected Circuits

Based on our preliminary examination of the optical shuffle, we find that the most attractive implementation for the near term involves the use of polyimide waveguides on both the boards and the backplanes, and high density arrays of optoelectronic sources and receivers. The densities likely to be encountered in parallel interconnection within massively parallel systems would suggest that large scales of optical integration would be employed. The integration need not encompass the associated drive and control electronics, but would merely be of a scale sufficient to allow the required interconnection density at the board edge. We emphasize that our choice of implementation was strongly influenced by considerations of practicality. Further developments in technology may ultimately render the free space approach feasible. Also, we were motivated by the need to interface with electronic processors, since optical processing elements are unlikely to offer improved performance over their electronic counterparts for the foreseeable future.

Several key components were identified as requiring development before an optical network could be implemented for highly parallel computing. These were: high density optical sources, high density optical receivers, and optical connectors between the boards and backplane which would be present in a practical system.

Considerations of power dissipation, environmental stability, compatibility with electronics, and reliability led to the choice of electro-optic waveguide modulators for the optical source. To achieve the required densities, modulator arrays rather than discrete devices are required. These would be fed by a remote laser, either within or outside the physical boundaries of the system.

13.1 Multilayer Polyimide Implementations

Consider a shuffle-exchange interconnected network containing 1024 PEs on a total of 16 boards, each communicating with message widths of 128 bits. Half the total number of channels will be on the backplane in the region of the central boards. While noting that some optimised routing may reduce the number of crossovers encountered by a signal in a given waveguide, in the worst case, a waveguide may encounter perhaps 10^4 other waveguides in its path. Assuming that in the worst case all other channels are "on" (although this would load the shuffle exchange network unacceptably), the signal arriving in the unilluminated channel under consideration would be -10dB down on that of the other channels. This corresponds to the limit of tolerable extinction ratio and signal to noise ratio to maintain a bit error rate of 10^{-9}. Some enhancement of this margin would be preferable.

As will be shown shortly, gradient index lenses may be employed to transfer the signals from board to backplane and back. Efficient use of the lenses would suggest that multilayer polyimide waveguides should be employed. At present, such guides are not available with the high relative refractive index demonstrated at Honeywell to perform the abrupt, complex routing which will be required.

Optical receiver arrays are also key components for parallel computing. For eventual compatibility with electronics, GaAs receivers are being investigated. Before a design can be prepared, a detailed analysis of the error budget for the system, the optical losses, and power levels must be performed. This will commence shortly.

High-density optical board-to-backplane connectors are required for the implementation of a practical system. The connector must transfer light from the waveguides on a board to those on a backplane. The connector must fulfil this function after repeated removals and insertions of the board, while maintaining low crosstalk for individual guides.

The use of gradient index lenses, an established technology, was investigated for these connectors. it was found that a commercially available lens would be able to handle all the channels associated with one processor. The properties of such a connector are compatible with existing electrical connectors. Longitudinal alignment is not critical, however angular alignments require careful attention. A design for a practical connector will be prepared shortly.

13.2 Board-to-Backplane Connector Design

If high density interconnection networks are to be implemented in practical systems involving several boards and a backplane, some means of transferring light between the components is required. In a practical connector, issues of alignment and its repeatability must be addressed. Given our previous choice of polymer waveguides for the interconnection network medium a connector must be developed to transfer light between waveguides mounted on substrates mounted at right angles to each.

Several methods were considered using polymer waveguides of varying indices to transfer light unidirectionally when the two waveguides were brought into contact. However, most of these require maintaining high optical quality of the surface throughout successive reinsertion of the connector. To meet our stated goal of general compatibility with present day multiboard architectures, we felt this approach to be unattractive.

Our favored scheme is to use Selfoc connectors in conjunction with prisms. Reasons for using this technology are that attractive tradeoffs between alignment parameters may be made, and that the technology is already relatively mature. At the start of the investigation, it was hoped that available hardware may be used. In contrast to other applications of selfoc connectors in expanded beam configurations, where a point-to-point configuration is used, we are concerned with an N-point to N-point mapping. The study presented in this section is then concerned with the density of waveguide connections attainable with the technology, the alignment tolerances of the system, and its loss and crosstalk.

The connectors allow us to tolerate poor lateral and longitudinal alignment, but force accurate angular alignment to be maintained. This tradeoff, wile not perfect, is more amenable to board removal and insertion than other approaches.

An envisaged implementation of a board-to-backplane connector is shown in figure 13.1. The waveguides in the vicinity of the connector are aligned in a plane orthogonal to both the substrates. Redirection of the waveguides to implement the perfect shuffle may be performed on the backplane.

Data from manufacturers (Nippon Sheet Glass) suggest that an individual lens may be able to image the outputs from all 100 waveguides associated with one processor. The density attainable from an ideally aligned connector is affected by the resolution of the lens, and by the field of view. The latter affects the intensity of the image associated with each guide. An optimum size exists for the waveguides to be used with a specific connector. If the waveguide is smaller than optimum, the divergence angle of light emerging from the guide will be large and a smaller area of the lens may be used before vignetting occurs. If the waveguide size is larger than the optimum value, the divergence angle will be smaller, but the density of waveguides must necessarily be lower. The optimum size and hence density of waveguides may be evaluated for single-mode guides. As the width of the guide increases, the number of modes supported will also increase. While the resulting connection density will be less than optimum, considerations of alignment and scattering loss may dictate that multi-mode rather than single-mode waveguides are used. Since the modulator is a single-mode device, and the expansion to a lower loss multimode guide is performed in the polyimide, it would be possible to perform the mode expansion in the backplane, keeping the waveguides single mode at the transmitter connector. This would increase the density of interconnections attainable at the transmitter end of the interconnection network.

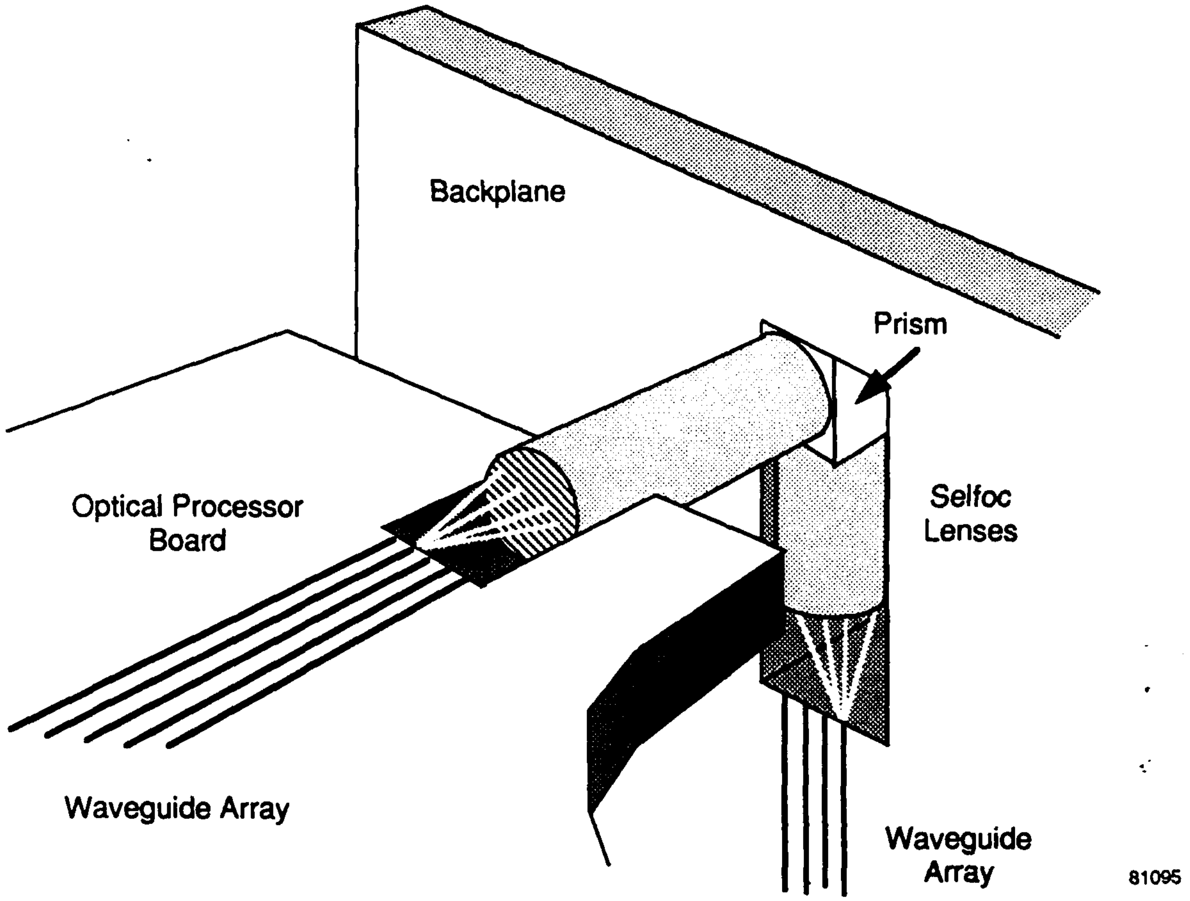


Figure 13-1. Board to Backplane Connector Concept

For a single-mode waveguide width of W, at a wavelength of 1 micron, the divergence angle of the assumed Gaussian fundamental mode is given by

$$O = \frac{4\lambda}{\pi W} = \frac{\lambda_e}{W}$$

We may define the limit to the field of view by the condition that the outer ray associated with the outermost guide barely touch the lens surface. If X_1 is half the field of view, then the number of waveguides in the field of view will be:

$$N = \frac{2X_1}{W}$$

Maximizing the resulting expression for the number of waveguides yields for the optimum waveguide width:

$$W_{opt} = \frac{f\lambda_e}{r_o}$$

And that the maximum number of waveguides which may be imaged is:

$$N_{max} = \frac{r_o^2}{\lambda_e}$$

The maximum number of allowable waveguides, expressed as a fraction of the optimum number versus the ratio of the guide Gaussian width to its optimum value, is shown in figure 13.2. This clearly illustrates the nature of the optimization to be performed.

As an example, for the NSG ILW lens family (65), with f/r_o = 2.623, the optimum value for the full width of the Gaussian is 2.77m and the divergence angle is 0.36 radians. For a diameter of 2.7 mm, the maximum resolvable number of spots would be 487. Allowing a separation of two spot sizes between adjacent waveguide images results in approximately 160 waveguides being used with this lens. As the waveguide dimension is increased above its optimum value, the divergence angle will at first decrease, then, as the number of modes supported increases, the maximum divergence angle for the highest order mode will remain at a value determined by the relative refractive index difference between the core and cladding.

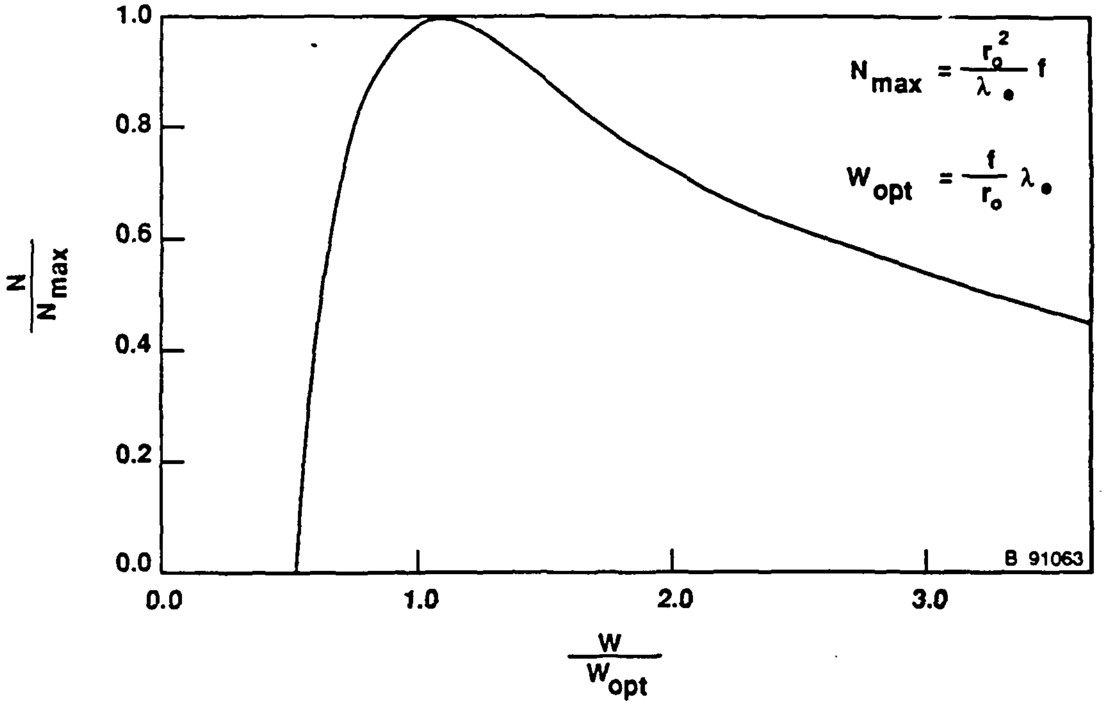


Figure 13.2 Resolution of optical system

Assuming that this angle is close to the optimum value for the single-mode case, a 10um guide would result in 240 resolvakle waveguide spots, and perhaps 100 waveguides being usable with sufficiently low crosstalk.

13.3 Tolerances in Selfoc-based Connectors

The polyimide interconnection medium itself is insensitive to small changes in wavelength. Since propagating light is contained within the waveguide, the direction taken by the light is determined by the defined waveguides. However, alignment errors in the connector may serve to image the light from one waveguide onto an incorrect location, and in severe cases onto the location associated with a different waveguide. It is therefore important to determine the tolerances for the complete system.

Two types of tolerance are important in the described connector. One set of tolerances refers to the alignment of the waveguides with respect to their intended positions in relation to the two parts of the connector. An example in this category would be a separation between the ends of the polyimide waveguides and the focal plane of the Selfoc lens. The second relates to errors in the alignment of the two connector halves. An example of a misalignment in this category would be the introduction of a gap between the two connector parts.

The alignment is insensitive to longitudinal misplacement between the two halves. If sufficiently severe of course, vignetting for the waveguides furthest from the center will occur. The greatest depth of field occurs where the beams are parallel, thus the 90 degree fold in the optical system should occur between the two lenses.

Of more concern are lateral misalignment and angular deviations. Deviations in angular placement of the two halves results in a shift of the imaged spot position. Lateral misalignment of the same parts results in a reduction in intensity for the outermost waveguides, but has little effect on crosstalk or resolution.

For a displacement of the receiving waveguide with respect to the imaged source waveguides, the crosstalk and optical loss which result are given by the appropriate overlap integrals [79]. The two are interdependent. Errors in placing the receiving waveguides within the connectors image plane are almost the same as the errors allowable for butt-coupling the waveguides. For multimode guides of 10 microns width, the lateral tolerances are of the order of a few microns. The longitudinal tolerances are calculated as follows. If the receiving guides are not placed at the focal plane of the second lens, the width of the beam will be larger then the value which would ensure optimum power transfer.

From [66], the fractional increase in beam size for a Gaussian of waist width 4 microns, and a wavelength of 1 micron, the tolerance is approximately 100um.

For an angular misalignment due to errors in mounting the prisms, the resulting misalignment is given the angular misalignment and the focal length. Thus allowing a 1 micron misalignment (single mode case) for the ILW lens discussed previously, the allowable angular misalignment would be approximately 2 seconds. As the ratio of the actual width to the ideal width (non-ideality factor) increases, so will the angular tolerance become less severe. Thus for the 10 micron guide, one might require angular tolerances of 10 seconds.

This trade-off of angular misalignment for longitudinal tolerance is beneficial in a system where boards are to be repeatedly mounted and demounted. A practical connector would maintain accurate angular alignment, while not requiring that the boards be inserted to a predetermined depth. The connector would then be compatible with electrical connector technology.

Figure 13.3 shows the concept of the board to backplane connector, including the mounting arrangement necessary to ensure repeatable, accurate angular alignment.

13.4 Fan-out Chip Design

The fan-out chip distributes the source power to a number of modulators. The length occupied by such a chip is of concern. For small fan-outs Y-junction splitters may be used [67]. The space occupied by the splitter depends on the minimum bend radius which may be sustained by the waveguides. The branch angle is typically limited to one or two degrees to avoid excessive transmission loss.

For high degrees of fan-out, the space occupied by the chip would become prohibitively large. A drastic reduction in length of the chip can be brought about be the use of waveguides incorporating right-angle bends. Associated with each such bend is a loss. Results reported to date for InGaAsP are 1.5dB per mirror [68]. For a fanout of 2^n,(n=2,4,6...) n such bends would be required.

Several options exist for the fan-out chip substrate. If the residual loss of the AlGaAs Waveguides is sufficiently low, it is possible that the chip could be fabricated from the same material. In this case, the fan-out chip and modulator could be fabricated on the same substrate, eliminating the optical loss and alignment difficulty associated with the interface between two chips. Alternatively, the fanout chip could be fabricated on a different material, enabling the waveguide design to be optimized for the two components independently. Constraints placed on the choice are that the single mode waveguides should match closely those of the

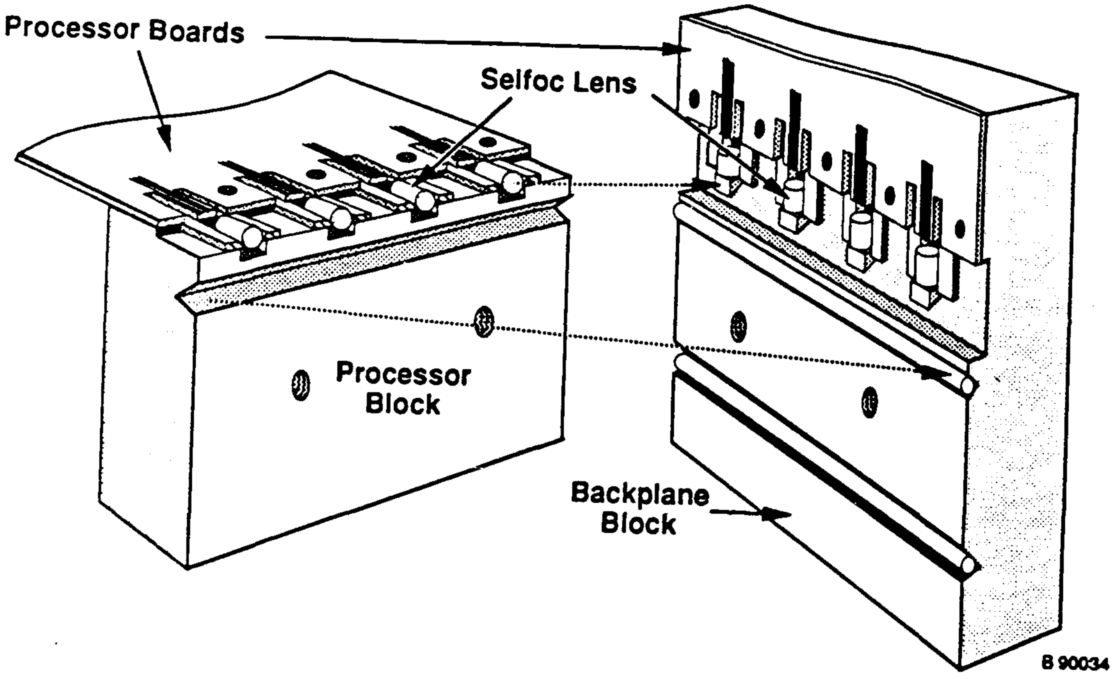


Figure 13.3 Multiple waveguide array board to backplane connector

modulator, and that the input sections of the fanout chip shɔuld be able to handle the undivided incident power. Chips based on commercially available ion-exchanged glass waveguides [69] would be suitable for hybrid integration with the modulators.

Since multimode polyimide waveguides have been identified as a suitable material for the interconnection medium, it would be advantageous to use the same materials and processes for the fan-out chip. Since efficient waveguide modulators will be single-mode in operation, the waveguides of the fan-out chip must be also. Results have not been reported on single mode implementations of this technology, and considerable development would be required to evaluate their potential.

14. Waveguide Modulator Development

To interface with our chosen polyimide waveguide interconnection network, in which the perfect shuffle mapping would be performed on the backplane, the modulators must be in waveguide form. To be immune to the variations in temperature likely to be encountered in such a system, and the variations in wavelength likely to be encountered if random source selection is to be employed, non band-edge type effects are to be avoided. The greatest immunity to these effects is likely to result from the use of devices which rely for their operation on the linear electrooptic effect. Several devices relying on such a principle have been reported, and are shown in figure 14.1.

Since most work on modulators to date has been on discrete devices, some effort was expended on selecting the device most appropriate for incorporation into high-density arrays. The devices illustrated in figure 14.1 may be divided into three broad categories on the basis of the fate of the unwanted light. Switches, both the directional coupler and the mode interference modulator, divert the unwanted light to a second output port. The functionality sought is that of a single channel whose transmission may be modified. Use of switches requires dummy waveguides to collect the unwanted light, thus placing further demands on the already stretched interconnection density.

Mode extinction modulators and Mach-Zehnder amplitude modulators force the unwanted light to become a radiation mode. While adequate for discrete device applications, the radiation may possibly re-couple to adjacent channels, or to other waveguides within the optical routing scheme. The usefulness of this class of modulators for array applications is not known.

The device which appeared to haver the greatest promise was an amplitude modulator based on polarization rotation and subsequent differential modal attenuation. Polarization rotators in AlGaAs/GaAs have previously been reported, but have had poor performance. The reasons for this inadequate operation, and our solution enabling the limitations to be overcome, will be described. A novel modulator configuration was developed, and shown to exhibit behaviour predicted by theory. The device was designed using a semi-vectorial finite-difference analysis technique, enabling the range of experimental parameters to be investigated to be kept to a minimum.

The two designs implemented were the polarization rotation based device and interferometric (Mach-Zehnder) modulator. Several techniques for isolating adjacent devices against undesired crosstalk were also investigated in order to determine the maximum possible density attainable with this technique.

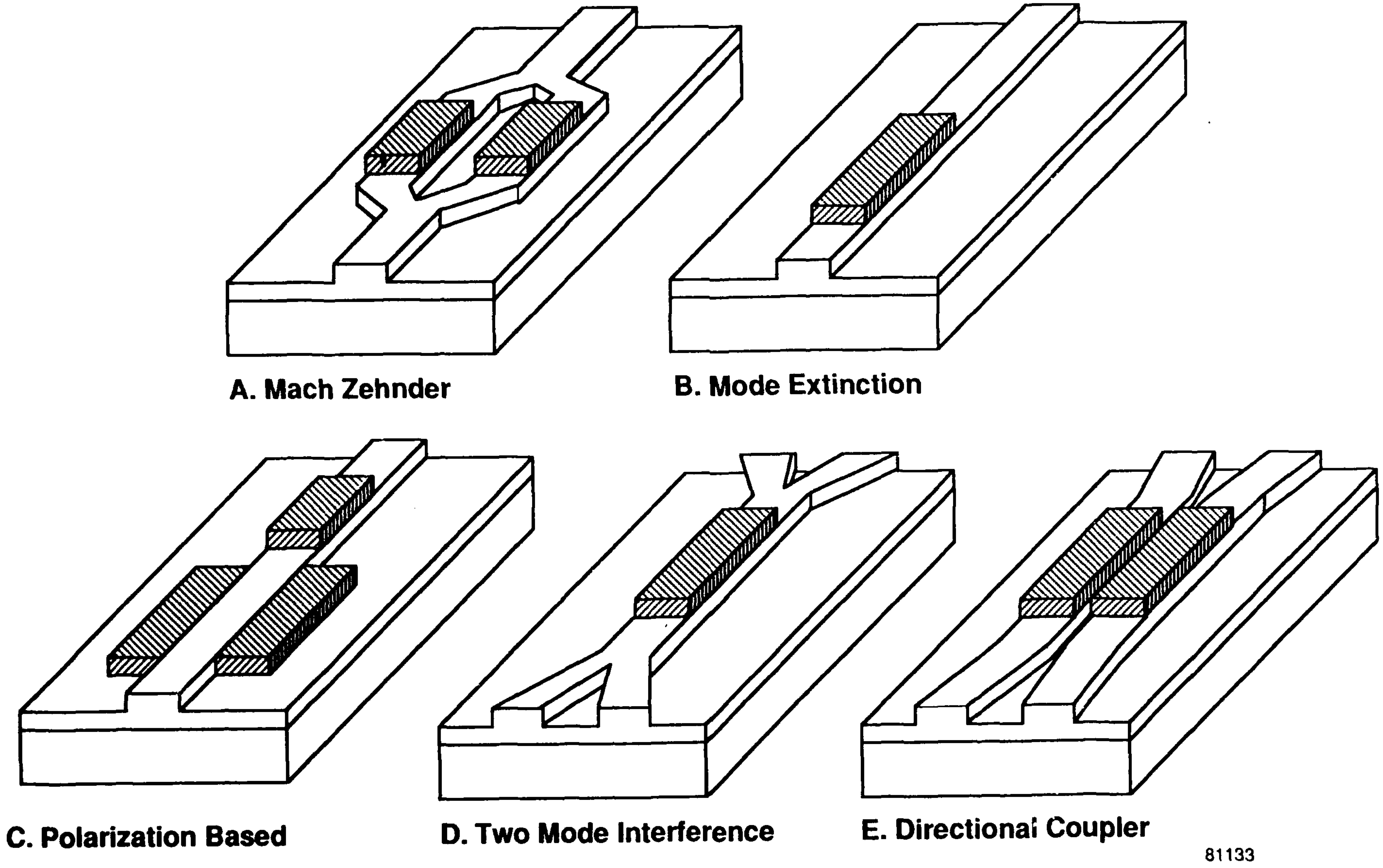
14.1 Modulator Design Choices

To render the modulator insensitive to variations in source wavelength or temperature of the system, devices based on the electro-optic effect rather than electroabsorption are preferable. In fact, several mechanisms may contribute to the modification of the transmission properties on waveguides. These are summarized in [70]. Our discussion is henceforth limited to a class of modulator which is minimally sensitive to changes in source wavelength and operating temperature. Five choices of electro-optic modulator are shown in figure 1. The designs are (a) a Mach-Zehnder interferometric modulator, (b) a mode extinction (cut-off) modulator, (c) a polarization based modulator, (c) an X-junction (two-mode interference) device and (e) a directional coupler. Further variations on these designs are made possible since different field orientations may be used with many of the devices. The following discussion is restricted to structures fabricated on (100) substrates. Electric fields along the <110> type directions result in a polarization type of modulation with twice the magnitude of devices based on <100> type fields. This is discussed by Wang in [78]. The <100> based devices may be operated in a phase modulation type of configuration, or also in a polarization type scheme. However, for (100) substrates the last option requires launching polarized light at 45 degrees to the horizontal and vertical axes of the waveguide. The difference between TE and TM propagation constants which exists in practical waveguides would severely degrade the modulator performance.

The Mach-Zehnder is an interferometric device. When used with electric fields in the <100> type directions phase modulation for the TE modes results. Most efficient use is made of the device in a push-pull configuration. The output of the single mode waveguide depends cosinusoidally on the induced phase difference between the two arms of the interferometer. Light arriving out of phase at the output waveguide junction is radiated into the substrate. This radiation is not of concern for discrete devices, however in high density arrays, the resulting crosstalk may degrade system performance. For n+ doped substrates, attenuation of the modes radiated into the substrate may limit the crosstalk sufficiently: further analysis is necessary to determine the magnitude of the effect. The device occupies approximately three "waveguide widths". The drive voltage is determined by the overlap of the electric field with the optical field, the refractive index of the material, and the electro-optic coefficient.

The mode extinction modulator or cut-off modulator is not an interferometric device. In this device, application of an electric field serves to modify the properties of the waveguide such that light is no longer guided within the structure [71]. The transmission therefore steadily increases or decreases with applied voltage reaching either a maximum or minimum value. The transmission function is not periodic with applied voltage. The device occupies only one "waveguide width". The unwanted light is

Figure 14.1 Electrooptic switches and modulators



A. Mach Zehnder

B. Mode Extinction

C. Polarization Based

D. Two Mode Interference

E. Directional Coupler

radiated into the substrate: further analysis is necessary to determine the resulting crosstalk in a high-density array. This modulator design with a <100> field configuration has the important advantage that the design is uncritical of fabrication parameters. The device may also be operated in the <110> electric field configuration. To gain an advantage over the previous configuration, the TE and TM propagation constants must be closely matched. A further benefit of this arrangement is that the mode orthogonal to the launched mode may be designed to be leaky. Thus the device would operate as a polarization converter between a guided and a radiated mode. The fabrication conditions for this device are more critical, in that control over the difference between TE and TM propagation constants is required.

The polarization based modulator operates in a similar manner, using <110> fields. Here the incident TM mode passes through the device unperturbed in the absence of an applied electric field to be attenuated by the polarizer at the waveguide output. The polarizer is a thin film device which relies for its operation on the selective coupling of the TM guided modes to a lossy surface plasma wave which may propagate at the boundary between a dielectric and a metal. Upon application of a suitable electric field, the incident TM mode is converted to a TE mode, which then passes unhindered through the polarizer section and onto the optical network.

The device requires horizontal fields if (100) substrates are to be used, and thus occupies rather more width than the mode extinction modulator with vertical field. However, crosstalk is expected to be lower since the unwanted light is attenuated rather than radiated into the substrate.

The two mode interference device, also known as a zero-gap directional coupler [72] relies on the excitation of the lowest order symmetric and asymmetric modes at the interface between a single-mode and a two-mode waveguide. The output of the device is determined by the relative phases of the two modes. Application of an appropriate electric field results in a modification of the phase delay, and hence results in switching at the waveguide outputs. The device may be operated in a normally on or normally off state, depending on which output from the device is selected as the output channel. The device occupies a width of approximately three "waveguide widths", and has a length dependent upon the difference in propagation constants of the two modes, and the required drive voltage. The device requires <100> type fields for operation in the manner described.

The directional coupler operates on a similar principle [73]. Two waveguides are brought into close proximity so that interaction occurs between the modes of the two waveguides. In fact the modes are strongly coupled, and device operation can be thought of as interference between the lowest order symmetric and antisymmetric supermodes of the complex waveguide. The device however requires that in the simple configuration illustrated, the interaction length is exactly one coupling length. Failure to arrange for this to occur results in reduced extinction ratio. The fabrication tolerances are relaxed if the device is operated in a more complex electrode structure, where the sign of the phase mismatch is reversed for half the length of the modulator. The requirement for interaction length of the modulator is now that it be between one and three coupling lengths for the waveguides, the coupling length simply being defined by the length required to transfer all the energy from one waveguide into the other.

The unwanted light for the two-mode interference device and the directional coupler is diverted to a "dummy" port. In both cases, provision must be made for the attenuation or diversion of the light to a "safe" location of the light. Thus these two designs are not favored due to their larger transverse size (three "waveguide widths" plus the space occupied by the optics required to dispose of the unwanted light.

The mode extinction modulator and the polarization modulator with polarizer both offer the possibility of operation in the normally off state, thus potentially eliminating the need for inverting circuits at the receivers.

Paramount in the design of the modulator array is the minimization of crosstalk between modulators. Intuitively, one would expect the polarization based device to have lower crosstalk in an array than the mode extinction device. In the latter, the light is forced away from the guide, and radiates into the substrate. In the polarization based device, the unwanted light is absorbed and converted to heat in the metal overlay. Isolation between devices may be accomplished in several ways. Constructing an "isolation trench" between devices may be effective, but requires the trench to be at least as deep as the gap between devices. Using conventional microfabricational techniques, this would not be practical. The approach investigated was to use isolation grounds between devices. This is effective, but may increase capacitance due to the proximity of a ground electrode to the drive electrode.

Modulation depth in the mode extinction device depends upon the voltage applied, and the induced conversion to radiated modes. The modulation depth of the polarization based device however is a periodic function of applied voltage, and is finite, limited by the difference in propagation constants of the two orthogonal waveguide modes. Evaluation of the feasibility of the polarization based modulator therefore requires an analysis of the optical waveguide to determine the propagation constants of the fundamental TE and TM like modes. Determination of the bandwidth in lumped electrode configuration requires detailed evaluation of the electric field resulting from application of a voltage to the device, and hence a determination of the capacitance.

Dissipation of heat in the metal overlay is of interest. Suppose a film of Aluminum 5 mm in length overlaid the output section of the waveguide, and that 1mW of optical power is to be absorbed with 100% efficiency with an average duty cycle of 0.9 (ie. almost always absorbing). With a waveguide pitch of 20um, a power density of $9W/cm^2$ would need to be dissipated. This is attainable with present day packaging technology.

The modulation depth of the polarization based modulator is affected by the polarization conversion efficiency, in addition to the extinction ratio (differential modal attenuation) for the polarizer located beyond the polarization modulator. Many choices are possible for the latter: the most compatible with planar technology would rely on the surface plasma wave excited at the boundary between a metal and a dielectric deposited above the waveguide upper surface [74]. Operation of the device is illustrated in figure 14.2. In the absence of an applied field, light launched in the TM mode propagates unperturbed through the modulator, and is incident upon the polarizer section of the device. Due to the selective TM attenuation of the device, the light is attenuated. If on the other hand an appropriate voltage is applied to the electrodes, polarization rotation is induced such that the TE mode is incident upon the polarizing section. Since little attenuation is induced for this mode, the light is transmitted. The extinction ratio of the device is determined by the conversion efficiency of the polarization converter, and by the extinction ratio of the polarizer.

14.2 Waveguide Design- Semivectorial Finite Difference Analysis

In order to design efficient waveguide modulators, some analytical framework is required. Modelling of the optical properties of such a device must address several issues. First, the propagation constants of the TE and TM -like modes must be determined. In the last few years, many analytical techniques have been proposed. Many models have been described in publications in the last two decades. Many of these are scalar techniques, in that a value of the propagation constants associated with a particular waveguide is determined with no regard for polarization sensitivity. This is a result of ignoring a term describing the gradient of the logarithm of the refractive index. Since the polarization properties of the guide are central to the operation of polarization based modulators, such techniques are not appropriate here.

Most versatile of the available techniques is the vector finite element technique. When applied to optical waveguides, realization of suitable resolution requires extremely long computer runs, so that simpler solutions should be sought whenever possible. The technique described by Stern [75] is a finite difference scheme for evaluating propagation constants and mode profiles of optical waveguides. It is semi-vectorial in that solutions to the eigenvalue equation are sought which rely on horizontal components of the electric field for TE modes being continuous across horizontal boundaries, with vertical components being continuous

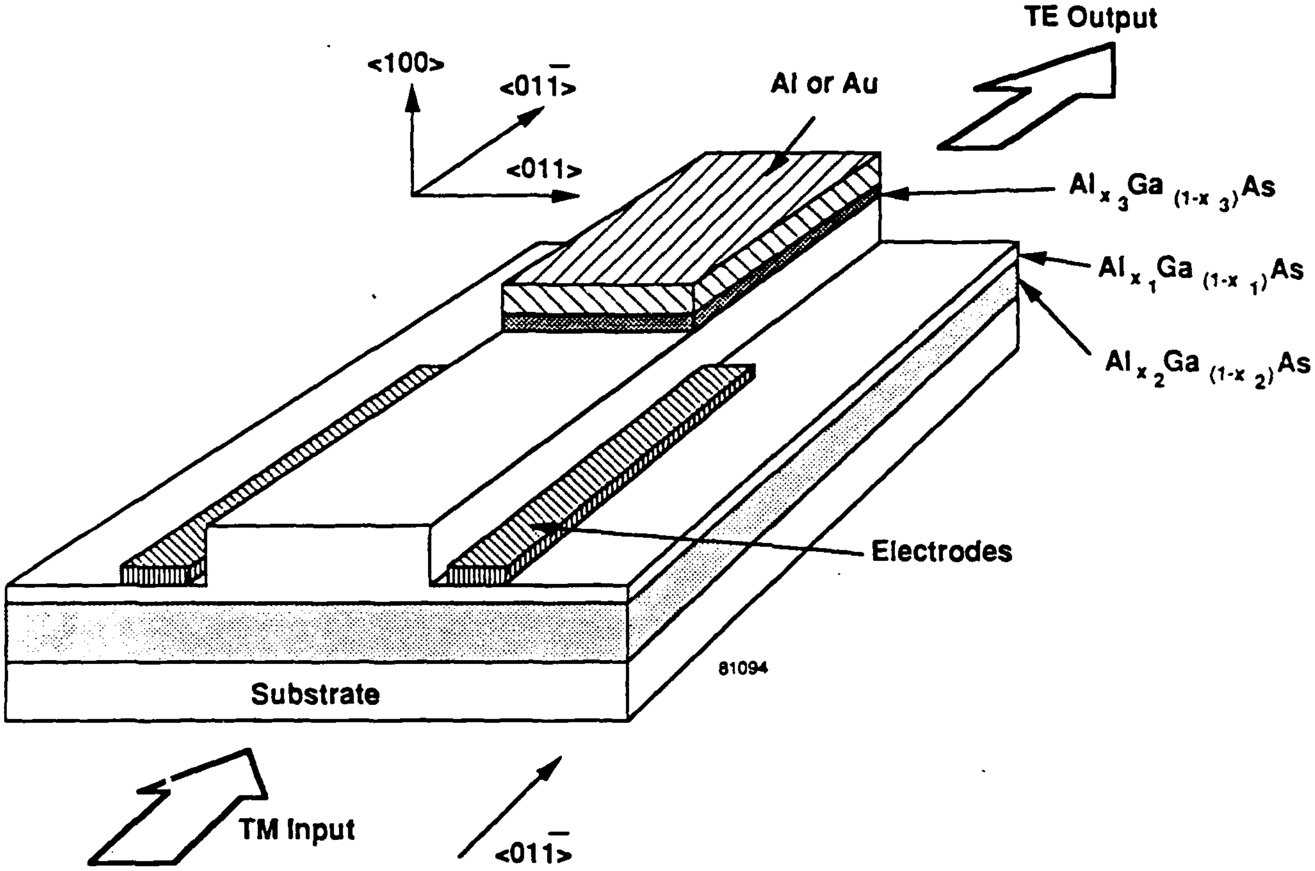


Figure 14.2 Polarization based intensity modulator

across vertical boundaries for TM modes. Justification for this technique is provided by the experimental observation that in the laboratory, waveguides typically maintain launched linear polarization over several millimeters. Indeed, waveguides which did not have this property would not be suitable for use in the systems under consideration here, since depolarization between the modulator section and the passive mode filter would result in both high excess loss in the off-state, and poor extinction ratio.

The structure analyzed is considered to be ideal in that no variation in parameters along the length of the guide is permitted. the model could however be applied to account for such phenomena if applied in a piecewise manner. The model deals readily with structures in which the boundaries are either vertical or horizontal. Structures incorporating interfaces at other angles may however be analyzed, providing the boundary may be represented by a series of horizontal and vertical boundaries. This limitation is also common to other techniques, including the finite element method, in which triangular elements are used to assemble the waveguide under investigation.

The method consists of solving the equation:

$$\nabla^2 E + k^2 E = B^2 E$$

to produce two sets of eigenvalues and eigenvectors, one corresponding to quasi-TE modes, the other to quasi TM modes. The initial N x N array of index values is represented as a 1 x N^2 vector. The finite difference matrix has dimensions N^2 x N^2

Using an appropriate algorithm, the eigenvalues of the matrix are returned as a 1 x N^2 vector. Further analysis of the results is then required to determine which of the eigenvalues corresponds to a valid solution of the field equation for the problem. For single mode guides, one solution only should be valid for each of the returned TE and TM eigenvalue vectors. For each eigenvalue there exists a set of N^2 data points constituting the associated eigenvector. For valid solutions, the eigenvector may be transformed into the N x N array of field values.

In the scalar case, an analysis of the waveguide is performed for real-valued refractive indices. The resulting finite difference matrix is both real and symmetric. A number of techniques yield solutions to the eigenvalue equation and return the associated eigenvectors.

Routines for finding eigenvalues of non-symmetric matrices are significantly slower that those for symmetric matrices. The run time would be further increased by the determination of the eigenvalues. Since the modeling is aimed at developing waveguides to have as little variation between polarization eigenmodes as possible, is was assumed that the eigenvectors for both TE and TM modes would depart little from the eigenvalues of the scalar problem. Thus the technique adopted was to determine the eigenvalues and eigenvectors of the scalar problem, then simply the eigenvalues for both TE and TM modes of the vector problem.

It should be noted that all results contained in this report, and obtained using the semi-vectorial finite difference technique were obtained assuming all indices were real. Further modifications of the program are in progress to incorporate imaginary refractive indices. This program takes 12 hours to run with the same resolution as the real version.

The scalar Helmholtz equation is solved using a finite difference technique. The physical N x N optical field values are written as a 1 x N^2 vector. The finite difference matrix has dimensions N^2 x N^2. The analysis proceeds by establishing the array of refractive index values, and using standard eigenvalue techniques to find N eigenvalues with their associated eigenvectors (the optical field values). Of the returned eigenvalues, only a few (two in the case of a single mode guide with non-degenerate effective indices) are meaningful solutions to the problem. Most eigenvalue routines take a time proportional to M^2, where M is the order of the matrix. Thus the finite difference method scales rather poorly as N^6. It should be noted that finite element techniques typically would scale as N^8.

14.2.1 Structures analyzed

Many semiconductor structures are possible candidates for the electrooptic modulator, and are illustrated in figure 14.3. Considerable work had been performed at Honeywell on high delta-n structures. This results in small mode profiles and small dimensions for single-mode operation. While these are not necessarily desirable properties, a further advantage is conferred on the waveguides by this structure, specifically that abrupt bends may be tolerated. Thus highly complex routing may be performed. This would be of use if the fan-out from the single laser source to a number of modulators were performed in the same waveguide system as the modulator. If however the fan-out is to be performed in a different system, the constraint of high relative refractive index difference may be relaxed. As will be seen when results of the modelling are described, the high relative refractive index structure possesses strong anisotropy in TE and TM propagation constants. Other structures with smaller relative refractive index differences are therefore also reported.

We shall show that the simple polarization rotator structure is predicted to suffer from poor extinction ratio due to imperfect phase matching between TE and TM propagation constants. These differences arise from geometrical effects, since the index material itself does not depend on polarization. Further modeling has enabled a modification to the structure to be designed. Specifically, a third electrode placed on the surface of the rib, and covering the entire width of the rib, has been added. The device now operates using two fields. A lateral field serves to couple TE to TM modes, while the residual asynchronism is removed, within limits imposed by device breakdown and acceptability of applied voltage, by virtue of a vertical field. The interaction of the applied electric field with the optical field is shown in figure 14.2. We note that similar principles, although with different field orientations, have been used in lithium niobate based devices .

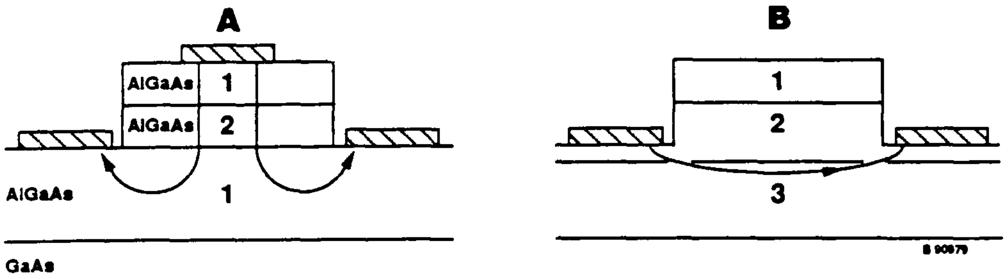
It will be observed that the electrode configuration for this device is essentially the same as for the phase-modulator type structures incorporated in the Mach-Zehnders. The difference between the two lies primarily in the etch depth of the structure.

14.2.2 Simulation Results

We present the graphical output from the numerical analysis of both the fully etches structure and the partially etched structure. It is important that the structures used for the fabrication of modulators support only a single spatial mode for both polarizations. For the strongly guided structures analyzed, the cutoff properties of the two eigenmodes are sufficiently close to each other, and to the scalar solution, that examination of the cutoff properties of the scalar guide will be sufficient. The semivectorial properties of our analytical tool are only employed when analyzing the polarization based structure, since then the modal asyncronism affects critically the extinction ratio and drive voltage.

Figures 14.4-14.7 show the lines of equal intensity of the scalar fields supported by a structure consisting of 1.05 microns of the 35% material, the same thickness of the 30% material, and approximately 3 microns of 35%AlAs concentration AlGaAs. The plots show the solutions for waveguide widths of 1.3 microns, 1.85 microns and 2.36 microns. The last guide supports two modes, the higher order mode exhibiting poorer confinement than the fundamental.

In figures 14.8 and 14.9 we show both the absolute value of the scalar eigenvalues, and the difference between the semivectorial eigenvalues of the two vector eigenmodes corresponding to each scalar solution. The onset of the second mode is clearly seen. Single mode behavior would be exhibited by guides with widths between 1.2 and 2.3 microns, with stronger confinement and hence less absorption by the substrate for the wider guides. The difference between the constants decreases with increasing width, as expected.



(a) Phase Modulator Structure

(b) Polarization Modulator Structure

Figure 14.3 Waveguide structures analyzed

Figure 14.4 Mode profile for 1.3um wide rib guide with full etch depth at 830nm wavelength

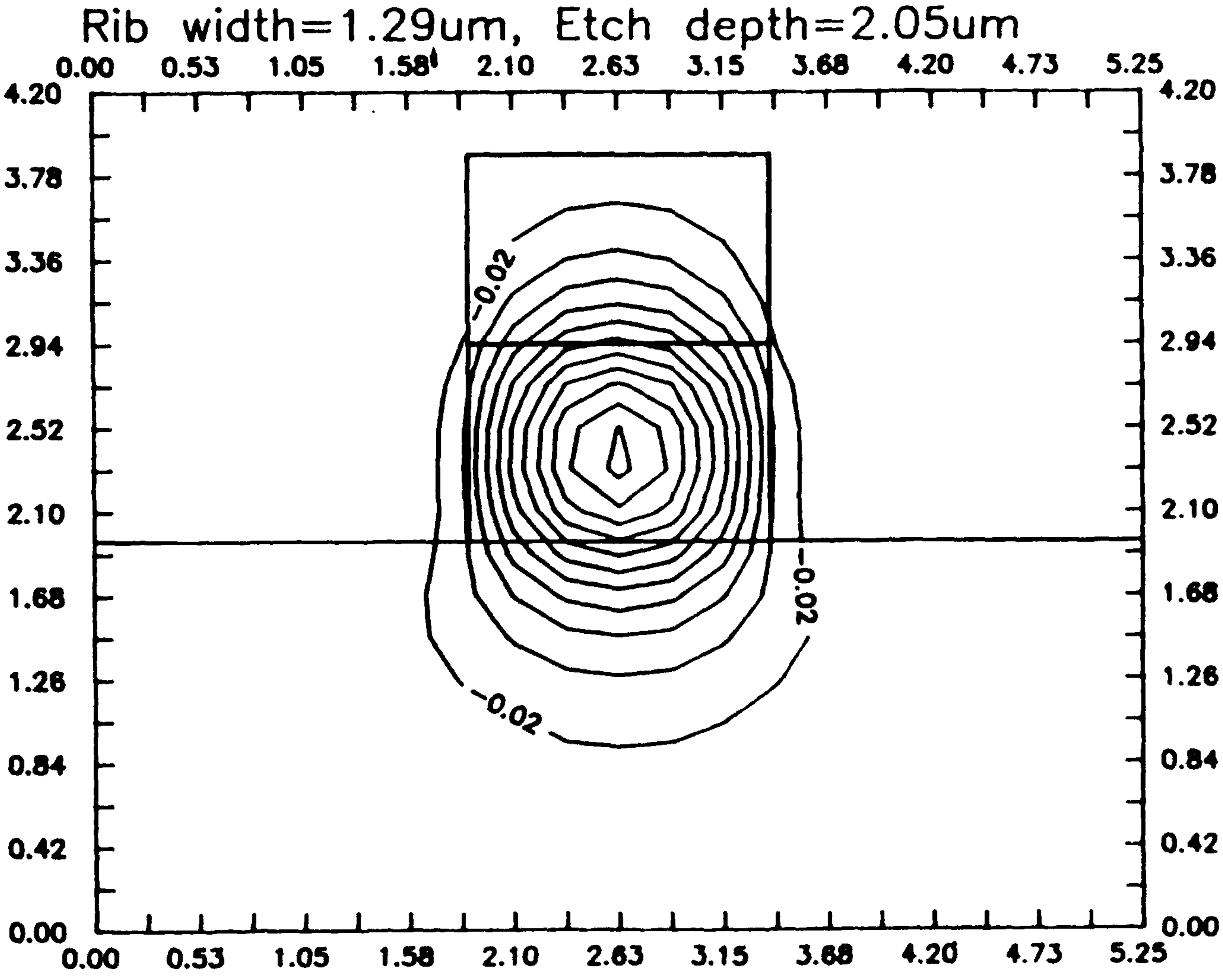


Figure 14.5 Mode profile for 1.85um wide rib guide with full etch depth at 830nm wavelength

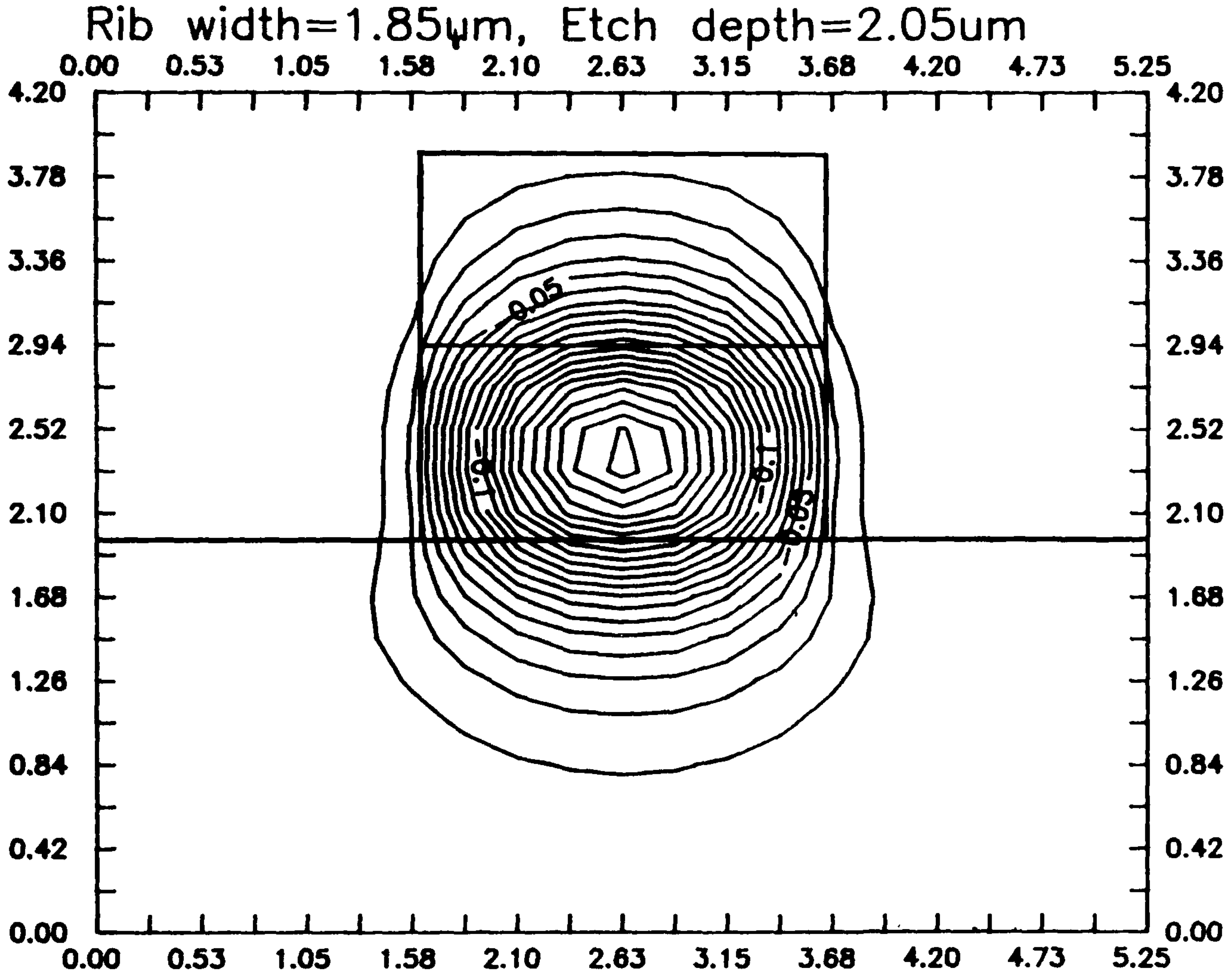


Figure 14.6 Mode profile for 2.36um wide rib guide with full etch depth at 830nm wavelength

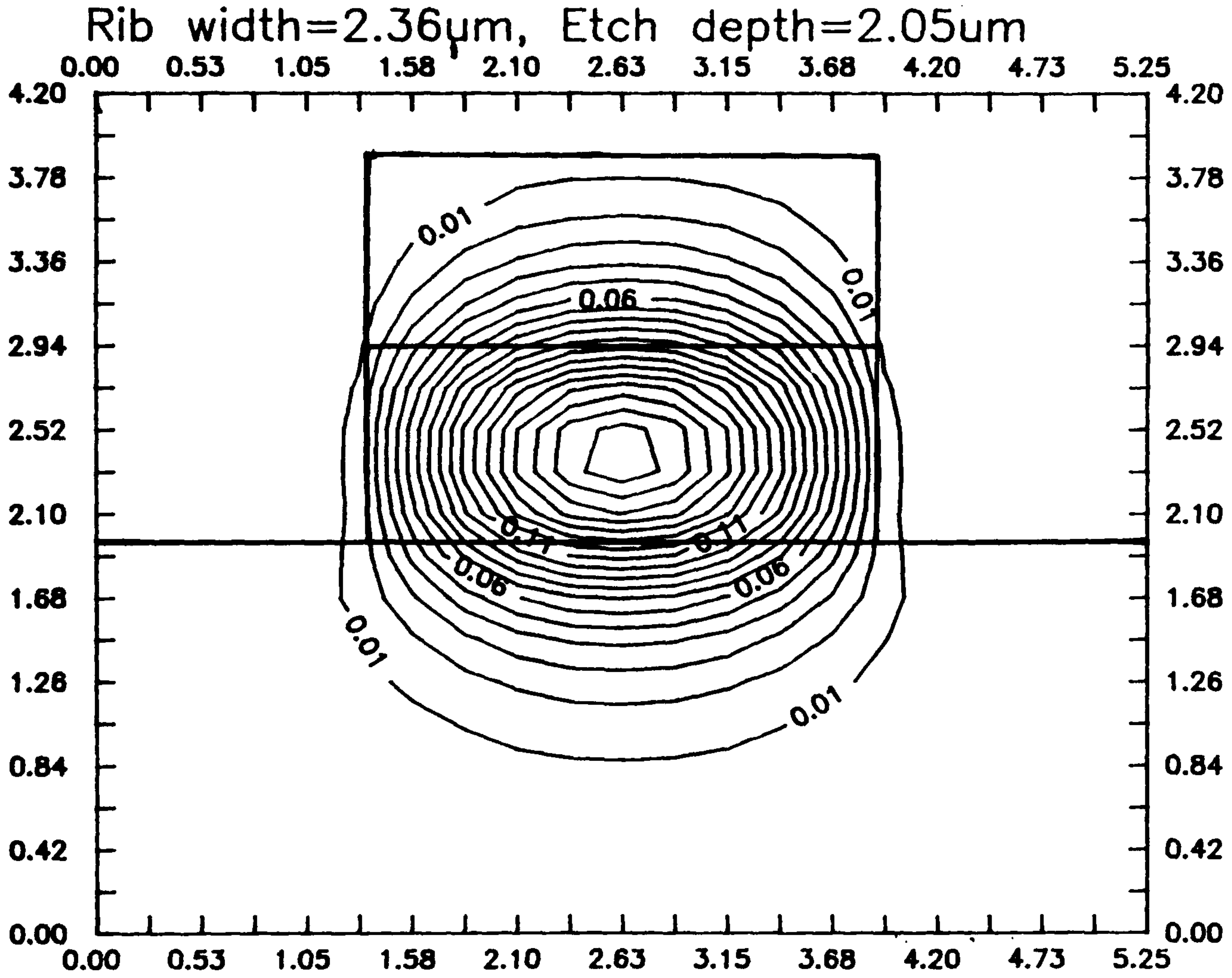
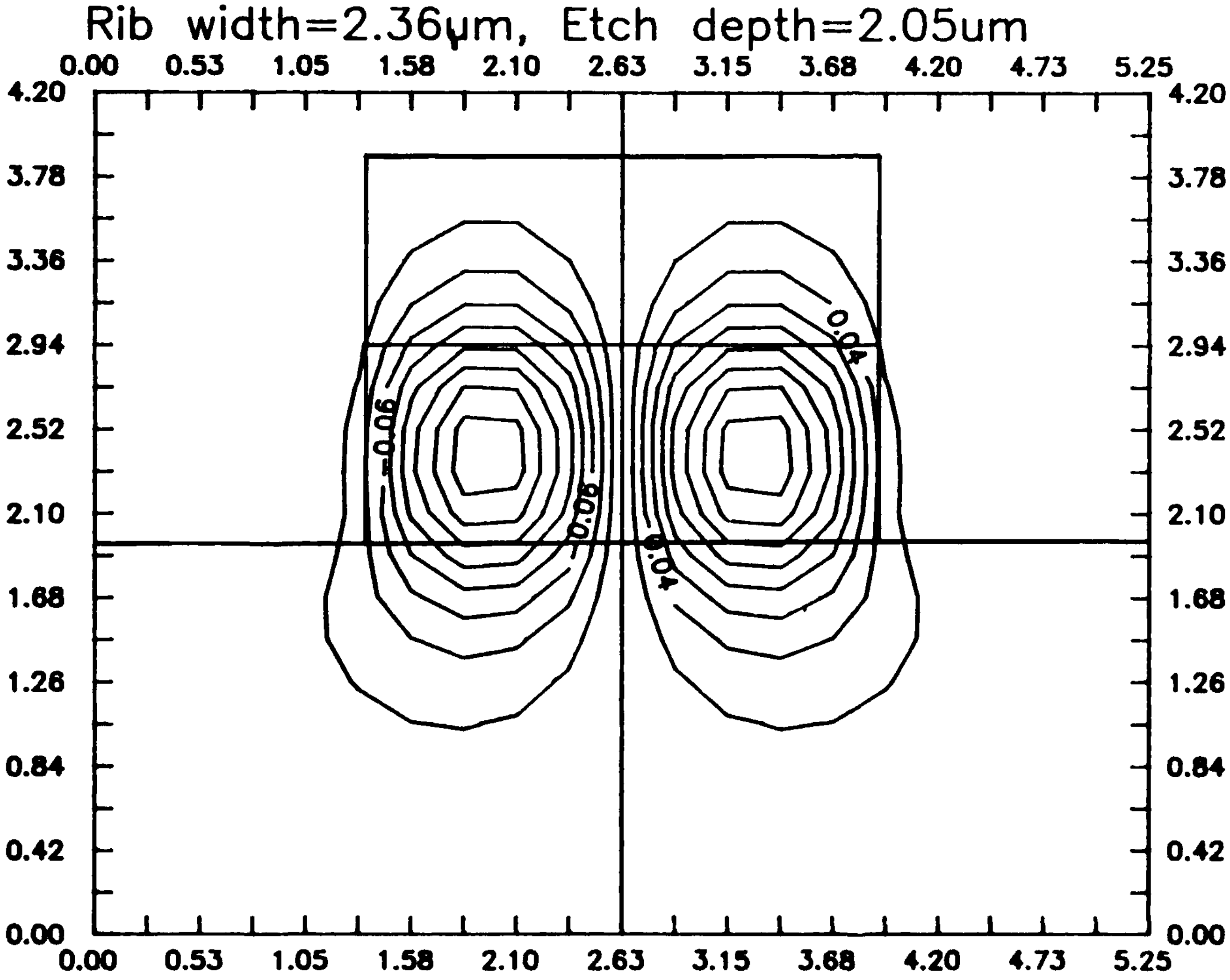


Figure 14.7 Higher order mode profile for 2.36um wide rib guide with full etch depth at 830nm wavelength



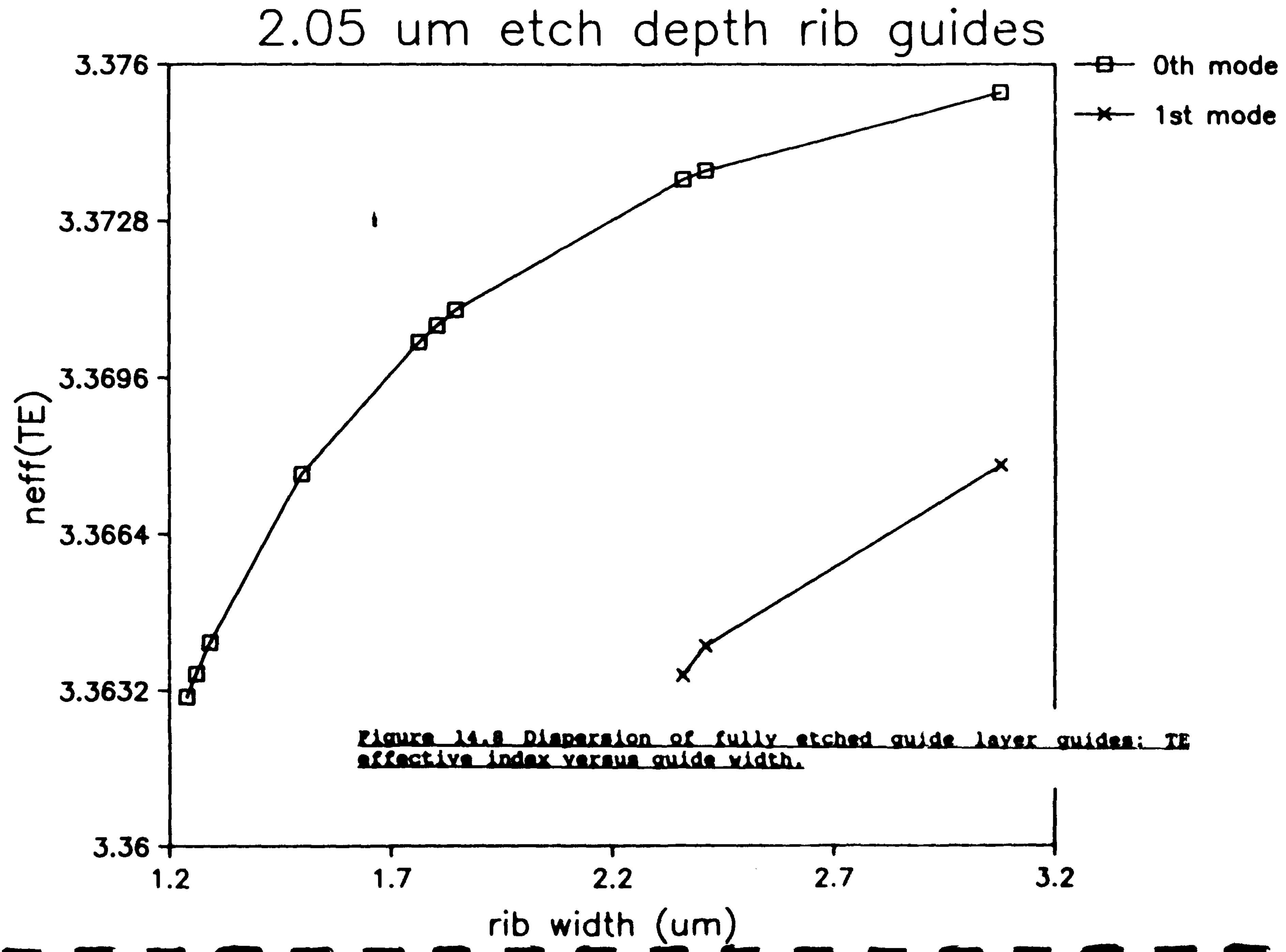


Figure 14.8 Dispersion of fully etched guide layer guides: TE effective index versus guide width.

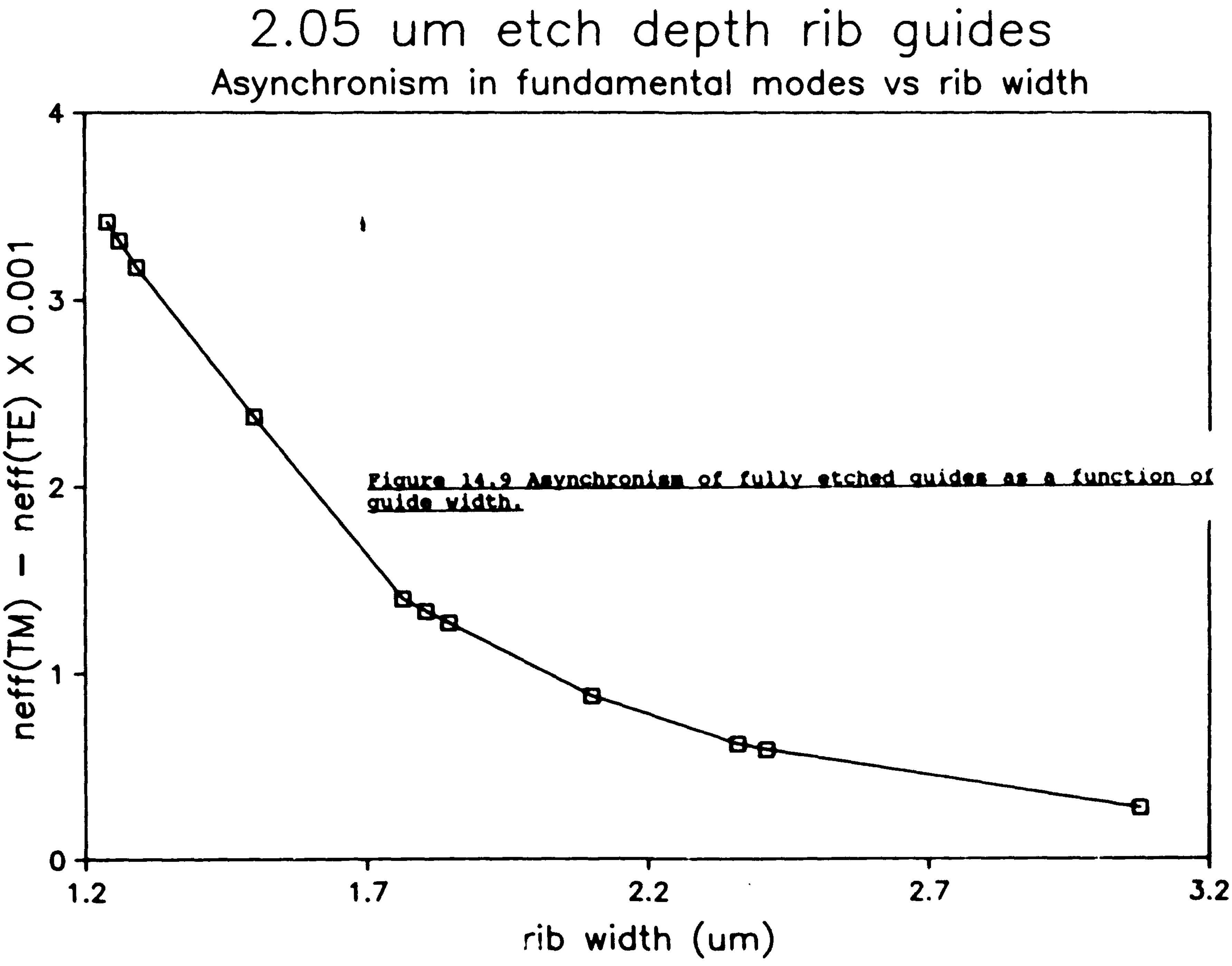


Figure 14.9 Asynchronism of fully etched guides as a function of guide width.

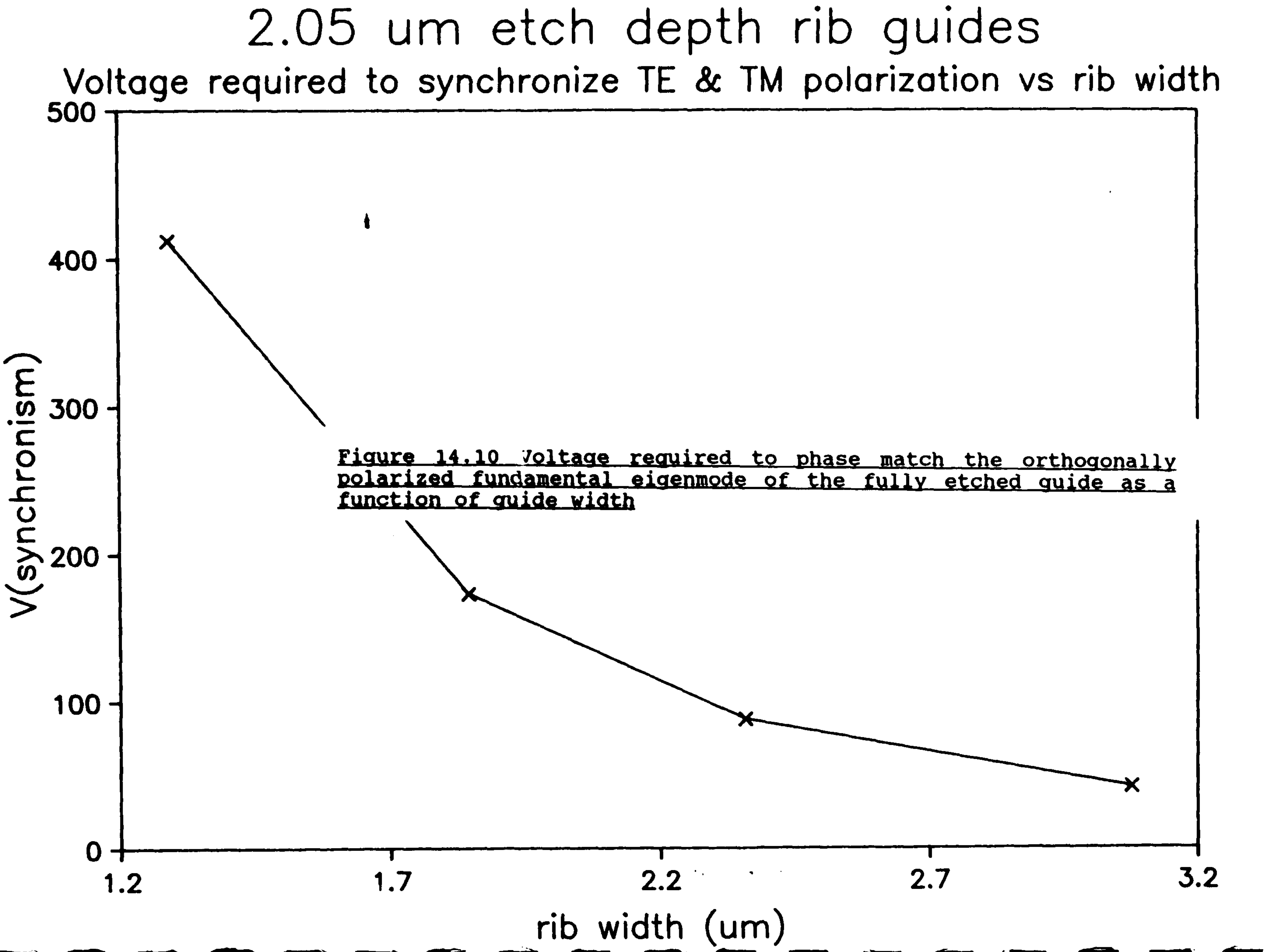


Figure 14.10 Voltage required to phase match the orthogonally polarized fundamental eigenmode of the fully etched guide as a function of guide width

In figure 14.10, we show the voltage required for a phase modulator structure to achieve synchronism. This was obtained using the electric field calculation routine, and determining the induced index change for a given mode as a function of applied voltage. The separation between rib and ground electrodes was assumed to be one micron. It was assumed that all charge had already been swept out from the guide. Note that the voltage required to achieve synchronism is typically several hundred volts. Both practical breakdown voltages, and the difficulty in obtaining high drive voltages experimentally from logic level drivers, dictate another approach. Figure NEW shows the variation in synchronism voltage with spacing between the rib and electrode.

Figures 14.11 to 14.21 show the mode solutions for guides in which the guide layer has been only partially etched. Poorer confinement is apparent in the mode profiles, particularly for the second and third modes, denoting by the first mode that with no zeroes in the guide. Figure 14.22 shows the dispersion of this guide as a function of width. Comparing with the results of the preceding analysis, it is seen that a narrower range of widths is permitted for single mode operation.

To operate a polarization modulator with either TE or TM input polarizations, phase matching is required. Figure 14.23 shows the degradation in attainable extinction which will result from a given difference in guided mode effective indices for the orthogonal eigenmodes. Since practical devices will only be able to sustain a certain voltage applied to a Schottky contact before breakdown occurs, it is important that the structure to be fabricated have an acceptably low breakdown voltage.

Figure 14.24 shows the voltage required to achieve synchronism for the partially etched structure. It is seen that for widths corresponding to single mode waveguides, the voltage required to achieve synchronism is approximately 20 volts.

14.5 Mask Design

The mask set designed incorporates the facility for the fabrication of several different devices. Specifically, we wished to fabricate Mach-Zehnder interferometric modulators, polarization rotators, and lastly a polarizer, or polarization analyzer. This is required for the polarization modulator, in which a polarization conversion device must ultimately be integrated with a polarizer to obtain amplitude modulation. In addition, however, we wished to maximize the number of measurements and hence obtain the most information possible from a given processing run.

Two sets of masks were fabricated. The first was designed to optimise the fabrication process and to evaluate and understand the performance of discrete devices. The second was intended to develop

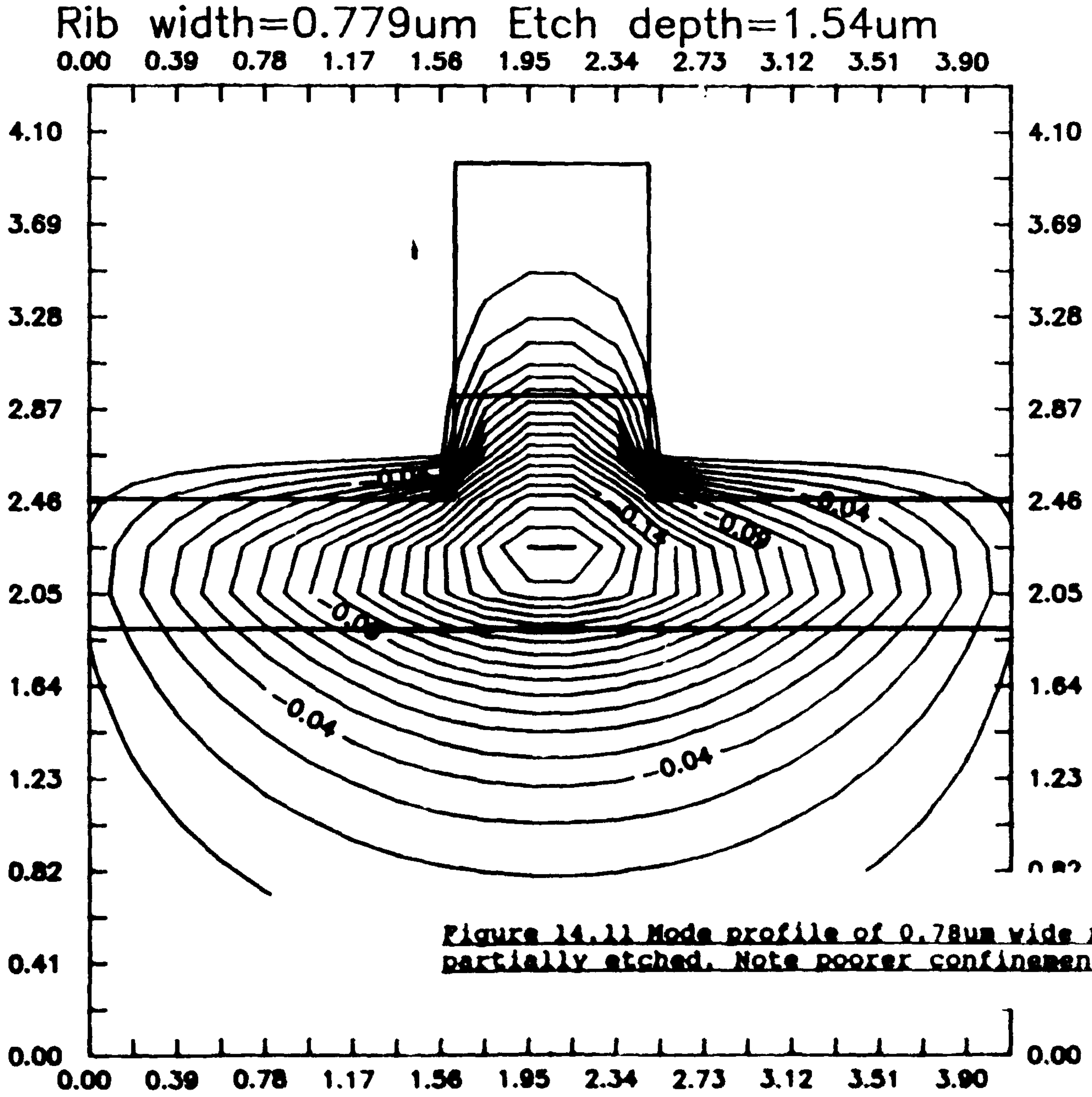
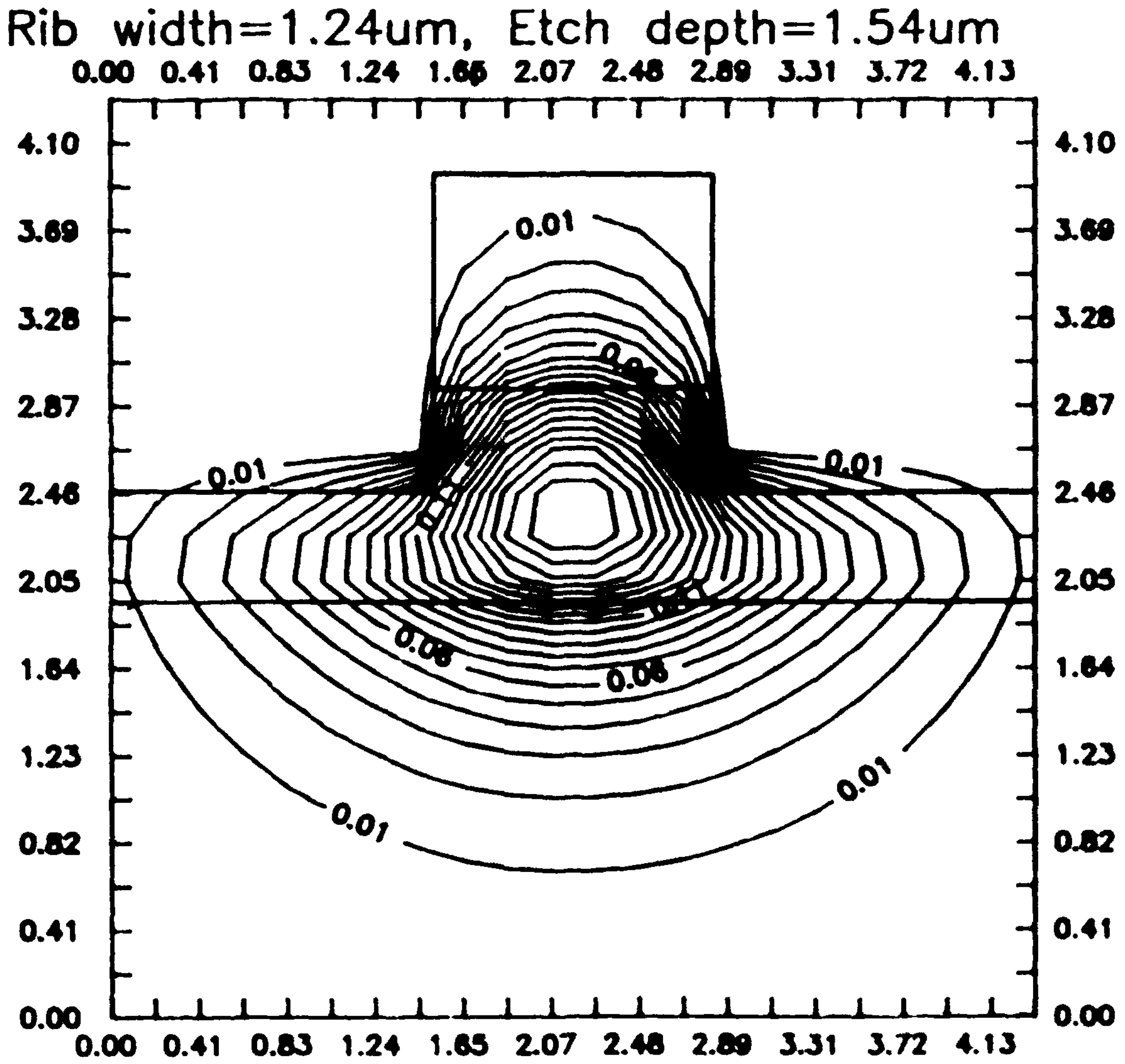


Figure 14.11 Mode profile of 0.78um wide rid guide with guide layer partially etched. Note poorer confinement factor.

Figure 14.12 Mode profile of 1.24um wide rib guide with partially etched guide layer.



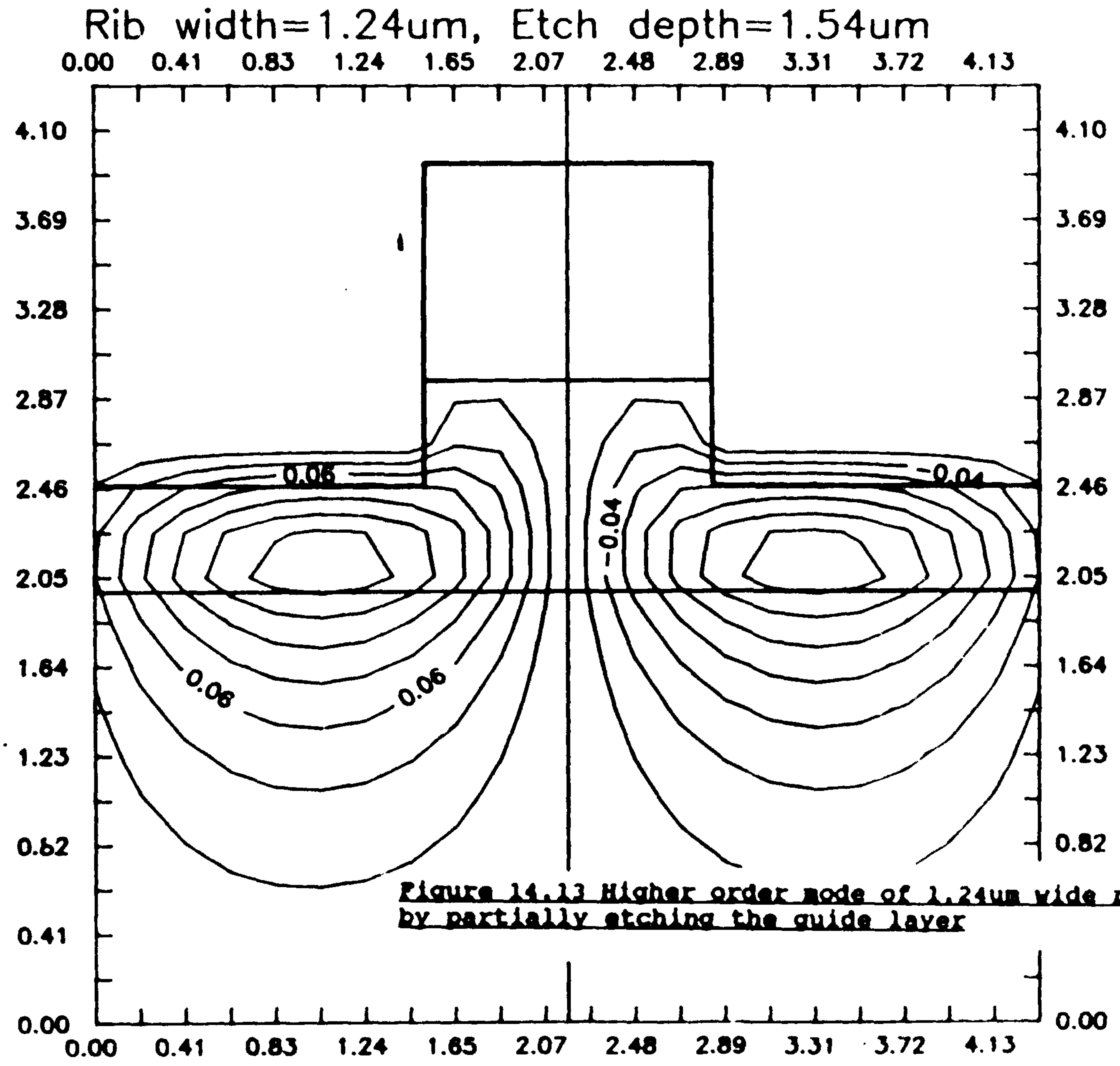
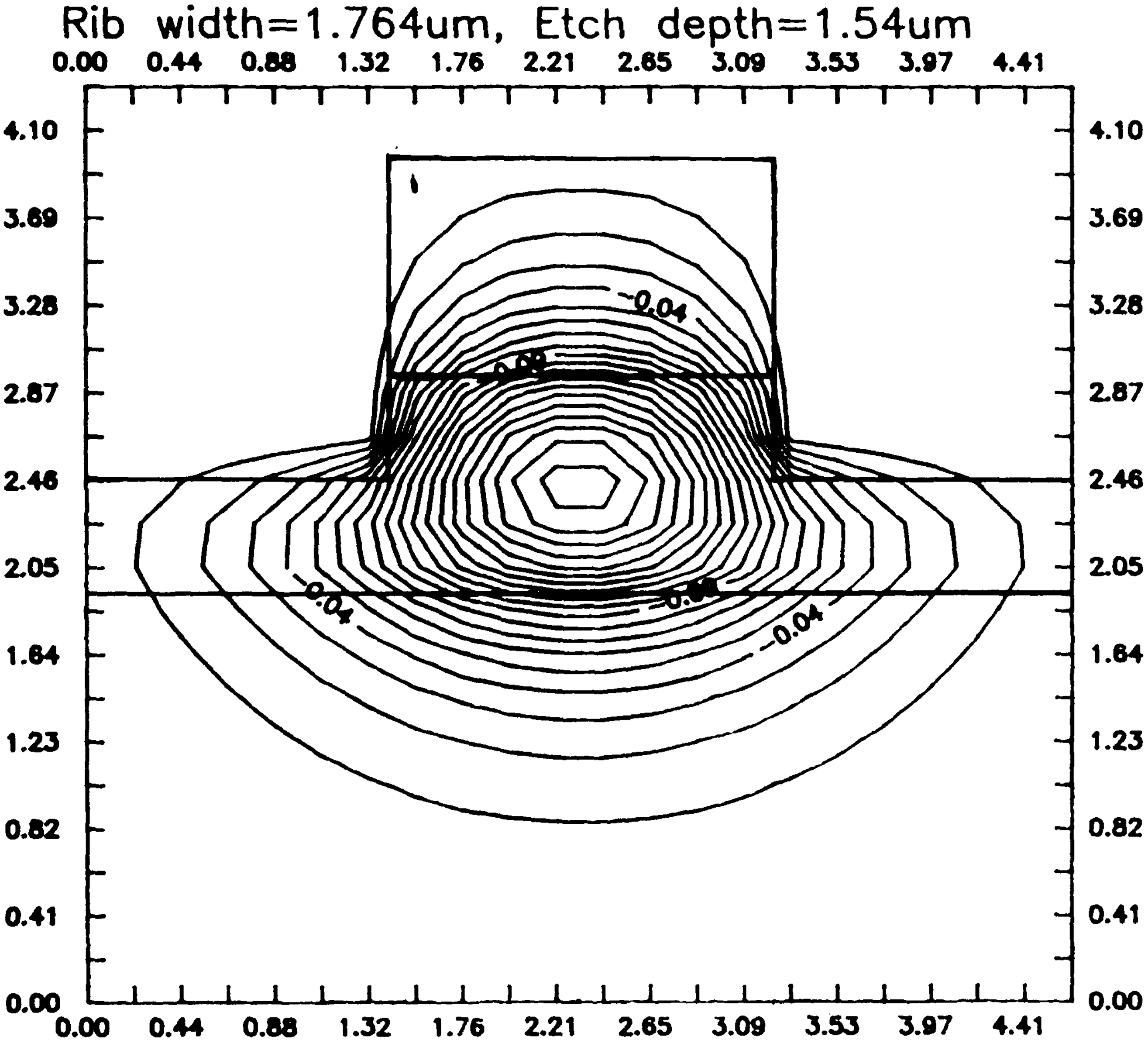


Figure 14.13 Higher order mode of 1.24um wide rib guide fabricated by partially etching the guide layer

Figure 14.14 Fundamental mode of 1.76um wide guide fabricated by partially etching the guide layer



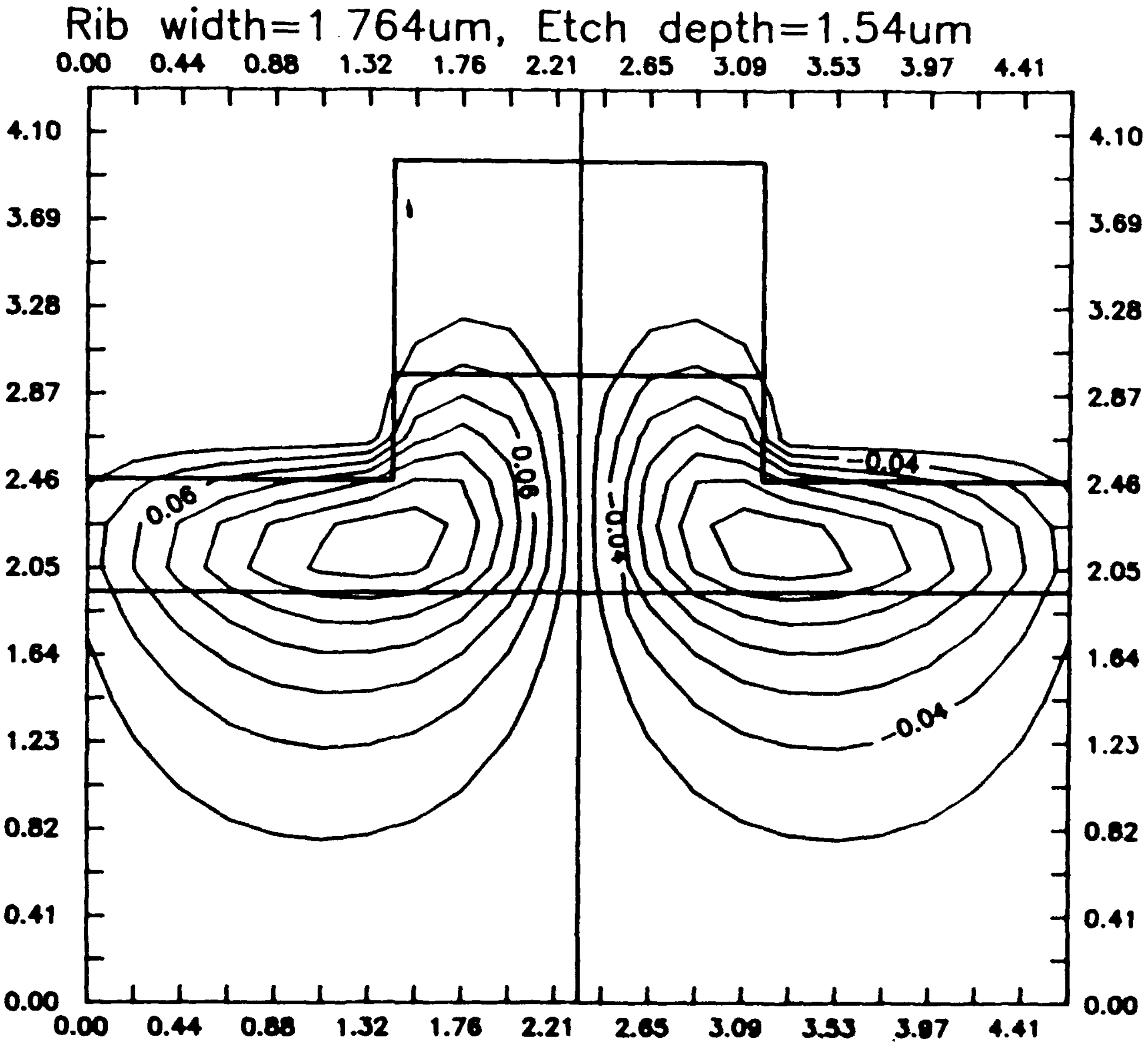


Figure 14.15 Higher order mode of 1.76um wide guide fabricated by partially etching the guide layer.

Figure 14.16 Fundamental mode of 2.435um wide guide fabricated by partially etching the guide layer

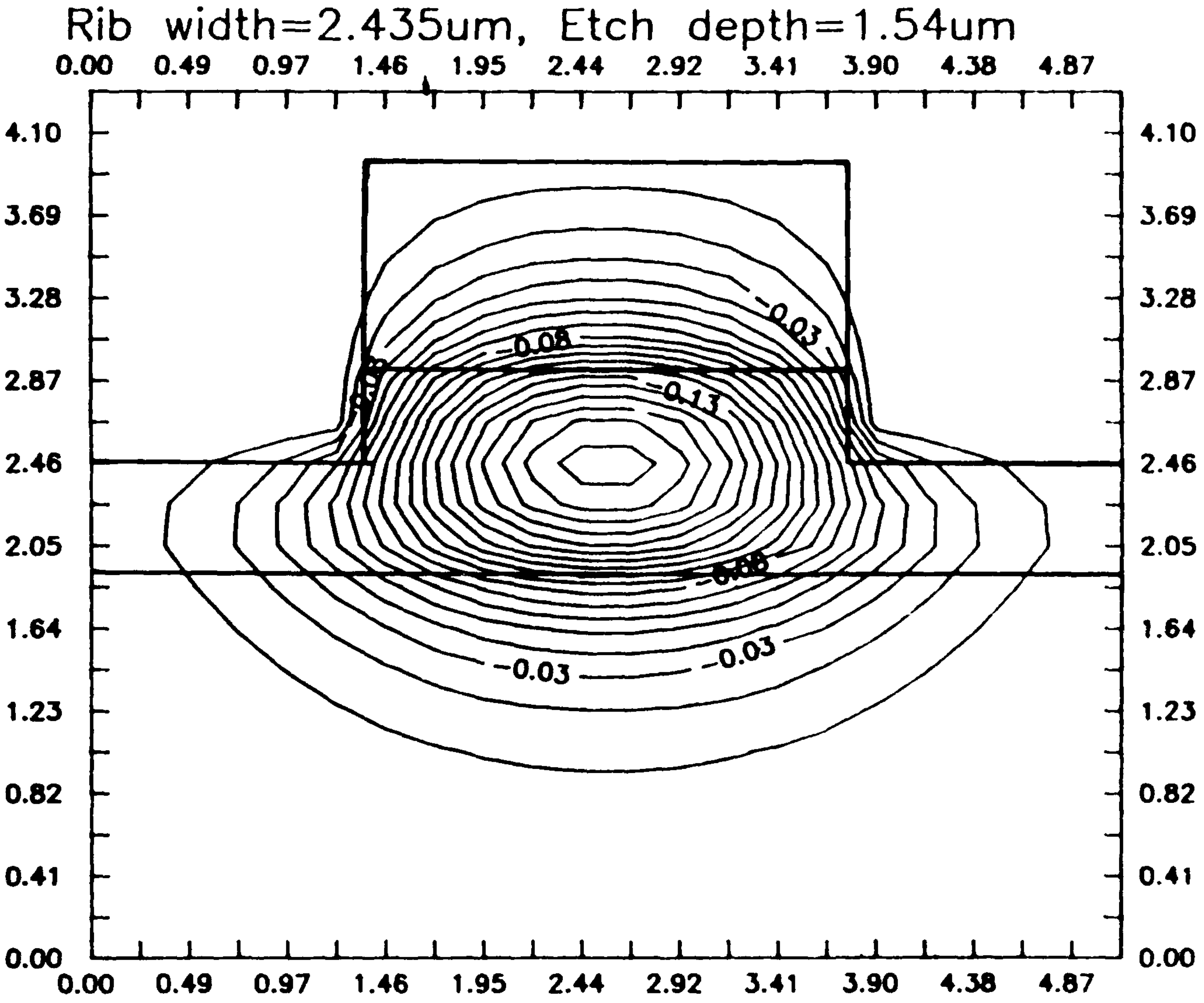


Figure 14.17 Higher order mode associated with figure 14.16

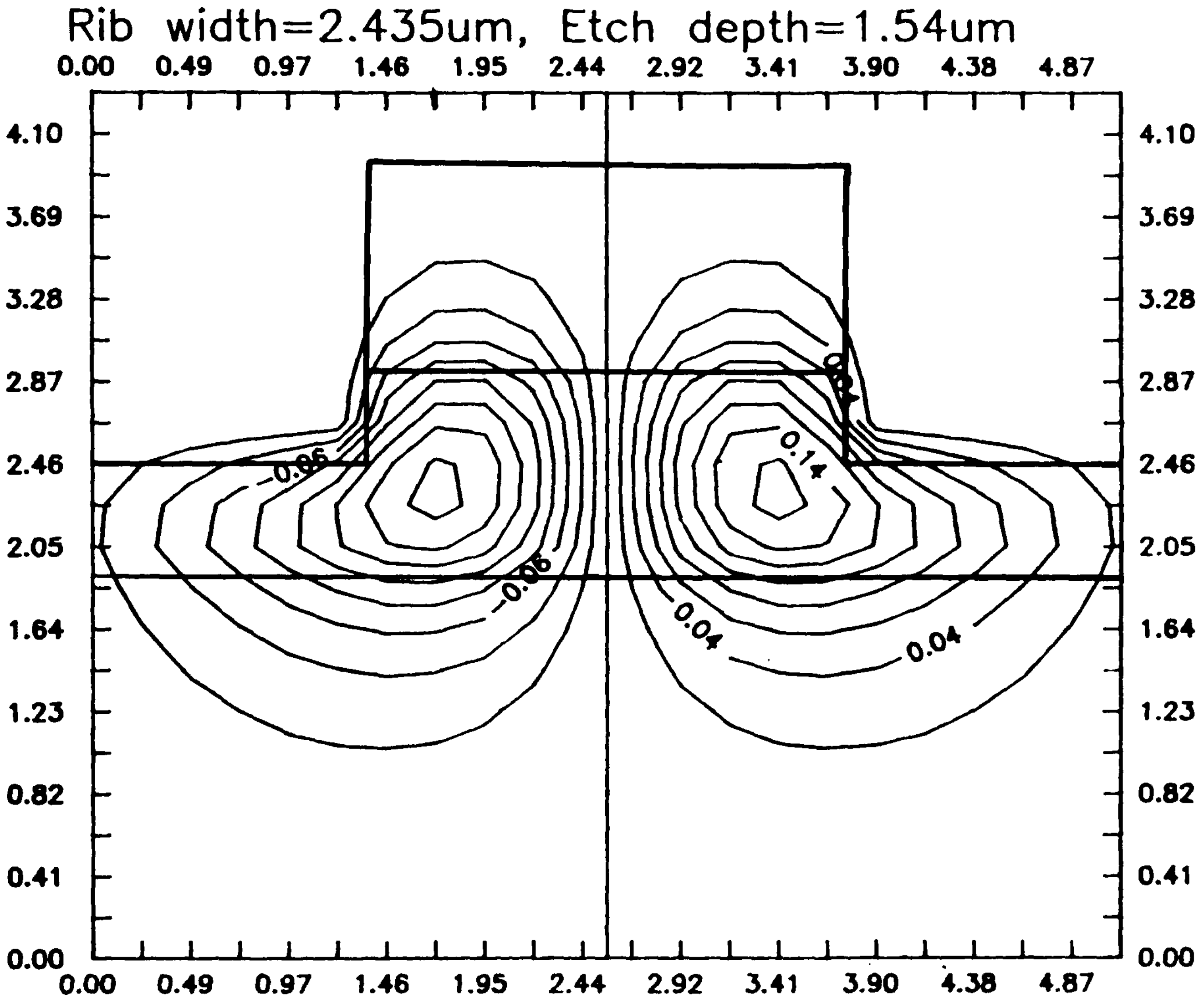


Figure 14.18 Highest order mode associated with figure 14.16

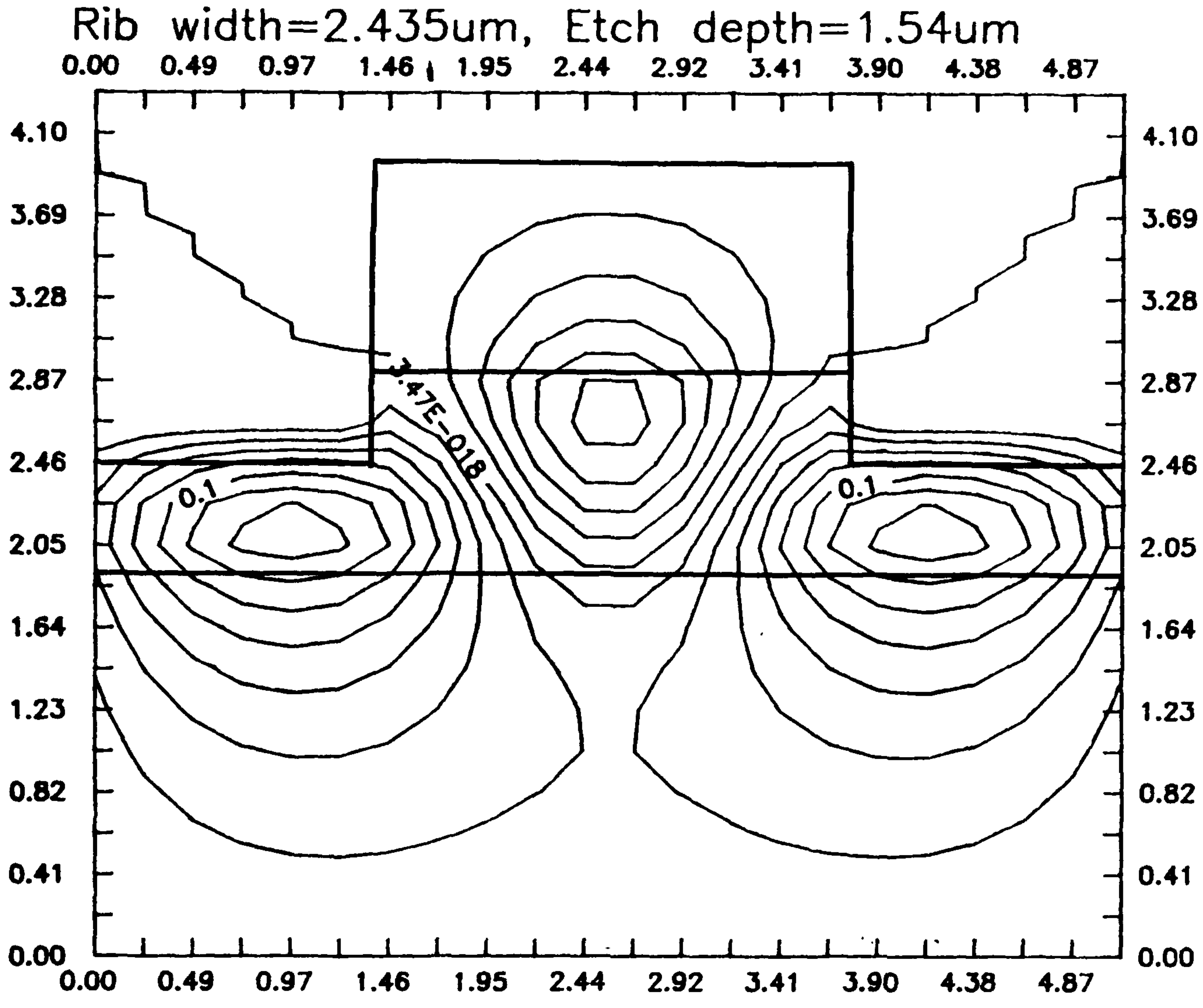


Figure 14.19 Fundamental mode of 3.07um wide guide fabricated by partially etching the guide layer

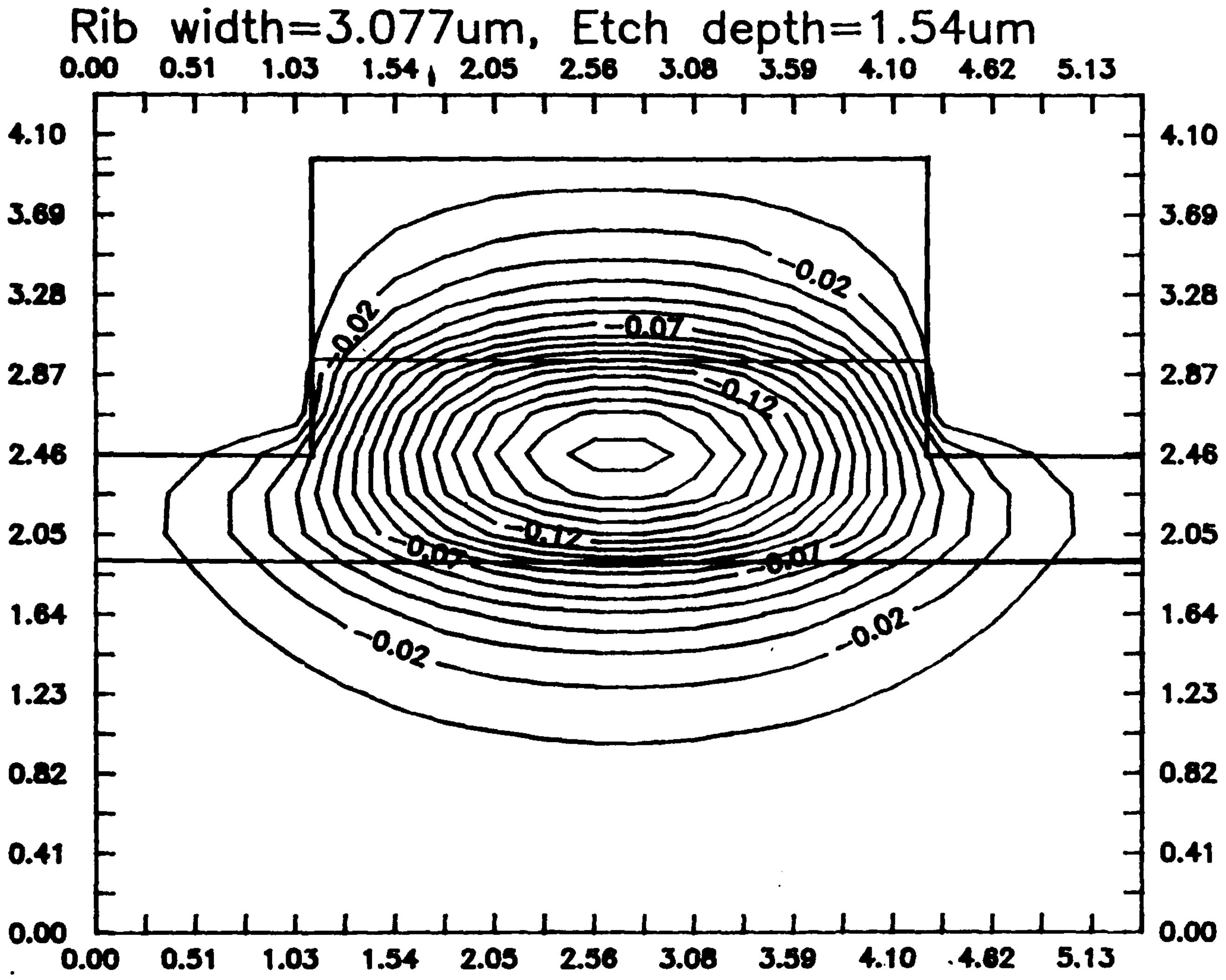


Figure 14.20 Higher order mode associated with figure 14.18

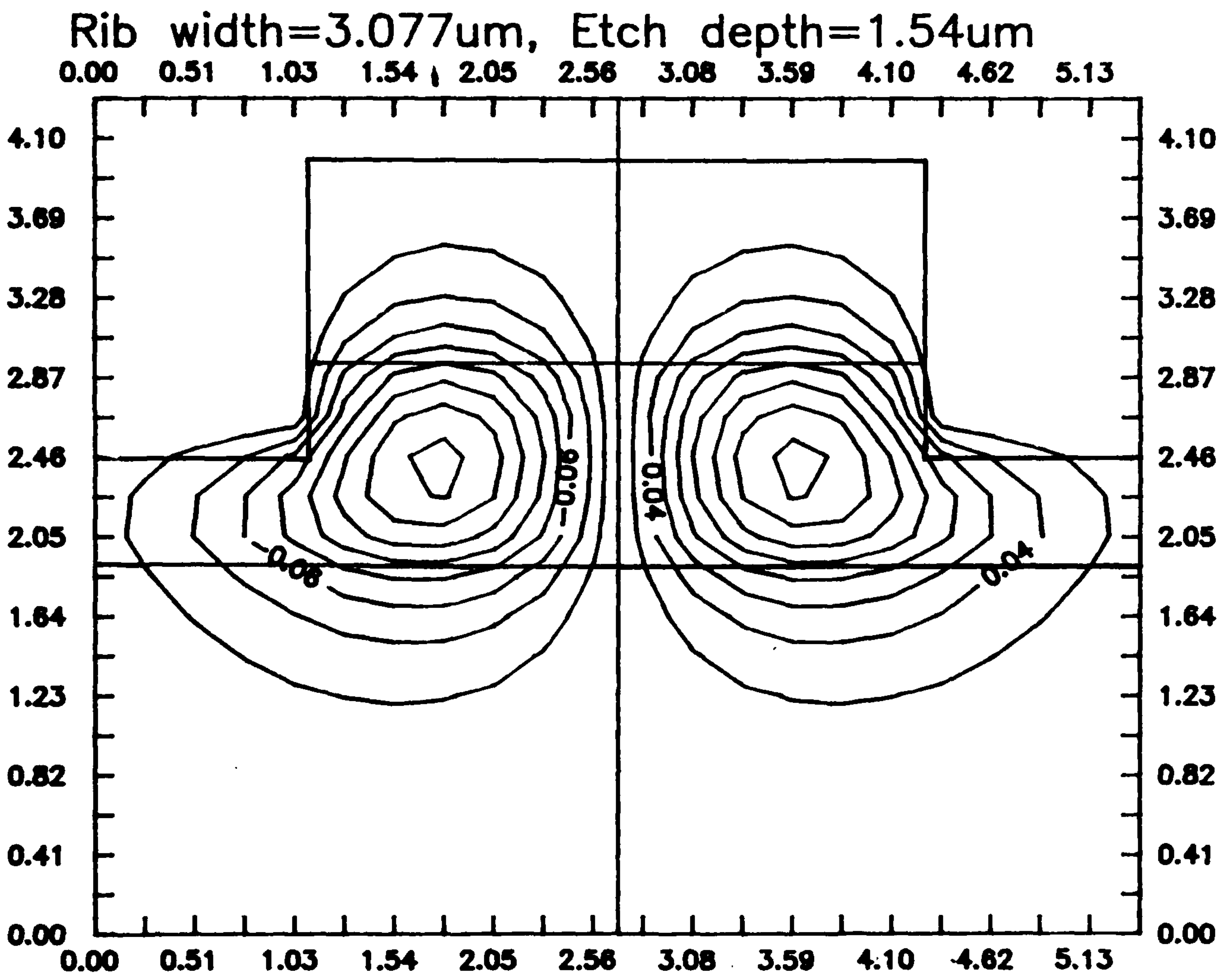
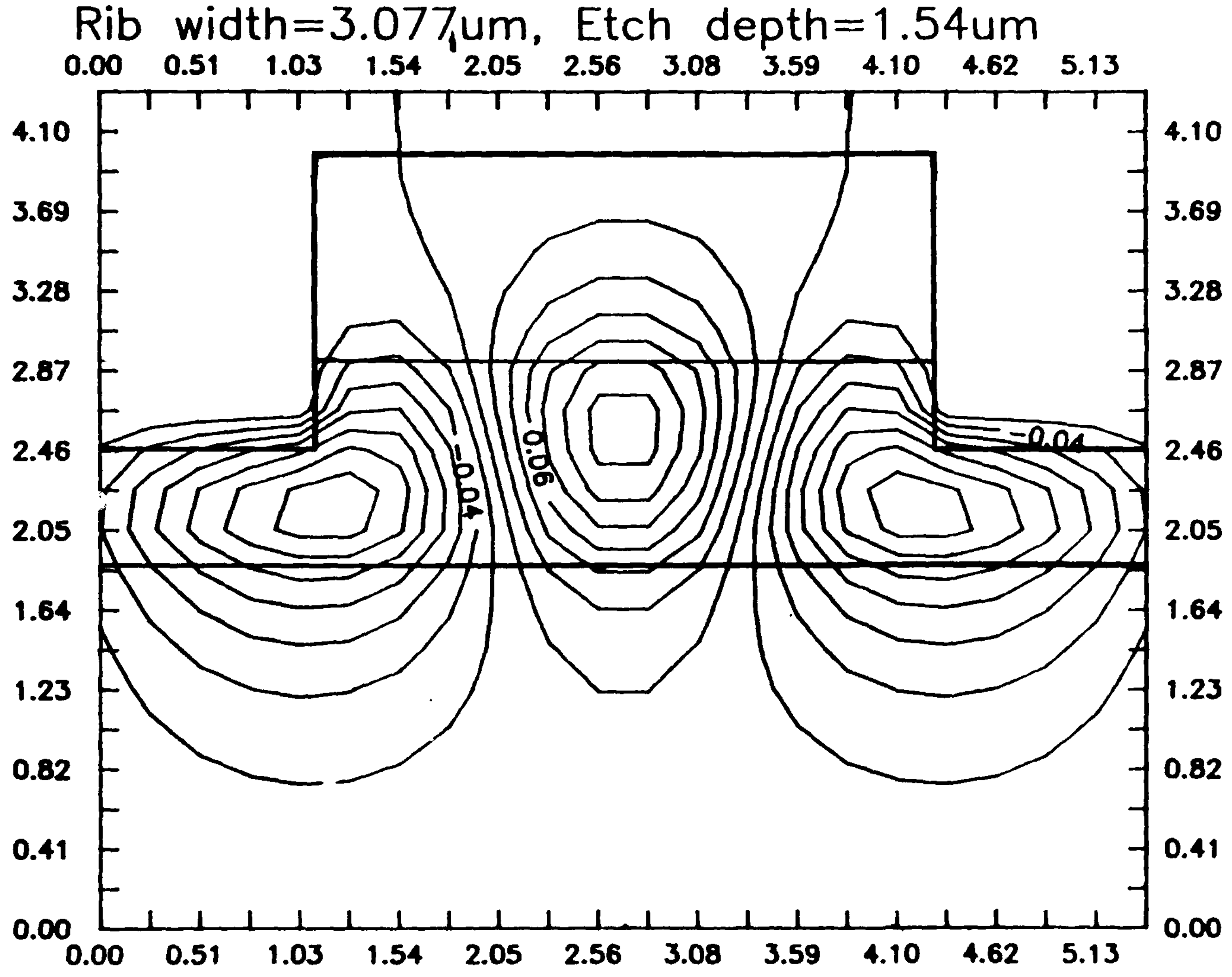


Figure 14.21 Highest order mode associated with figure 14.18



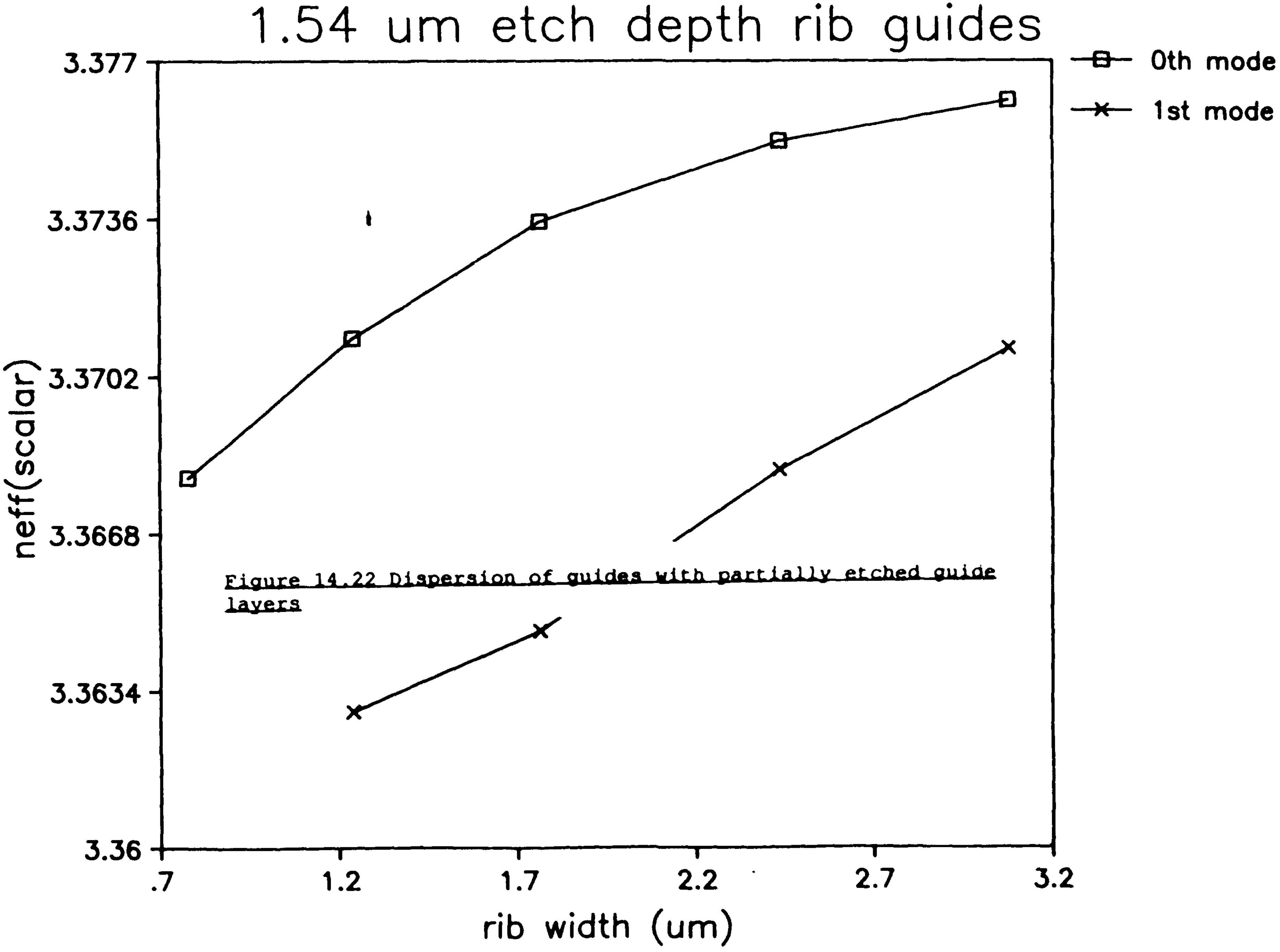


Figure 14.22 Dispersion of guides with partially etched guide layers

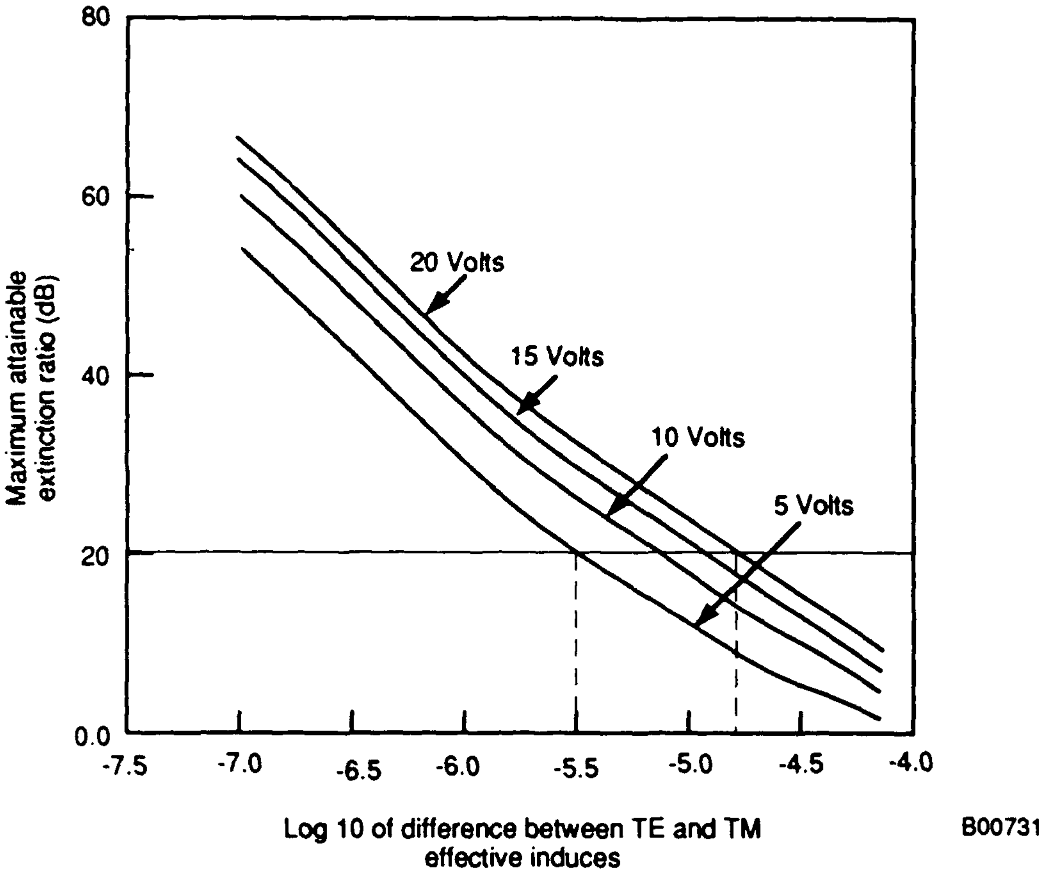


Figure 14.23. Maximum attainable extinction ratio for GaAs polarization modulators as a function of the asymmetry of the fundamental orthogonal eigenmodes.

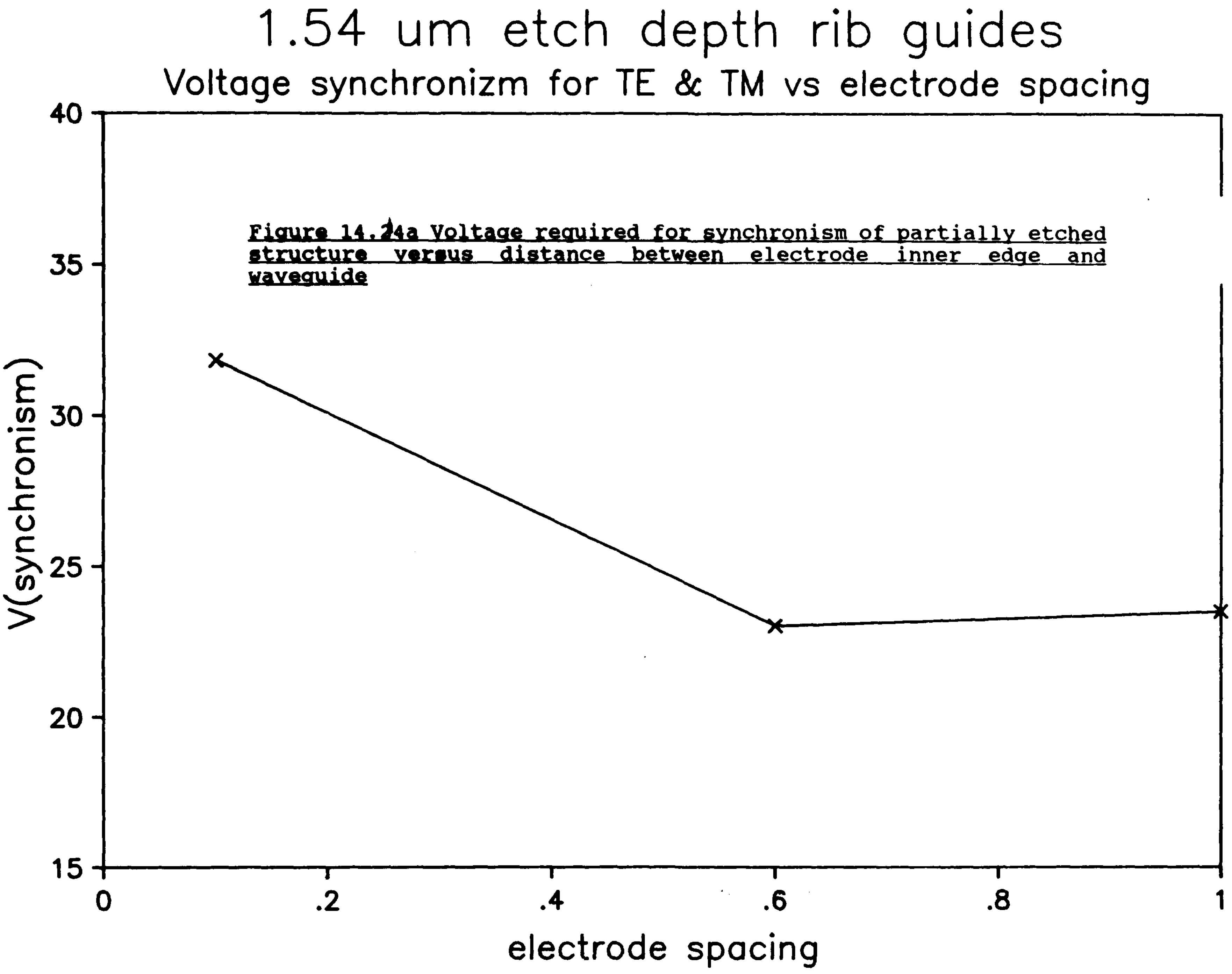


Figure 14.24a Voltage required for synchronism of partially etched structure versus distance between electrode inner edge and waveguide

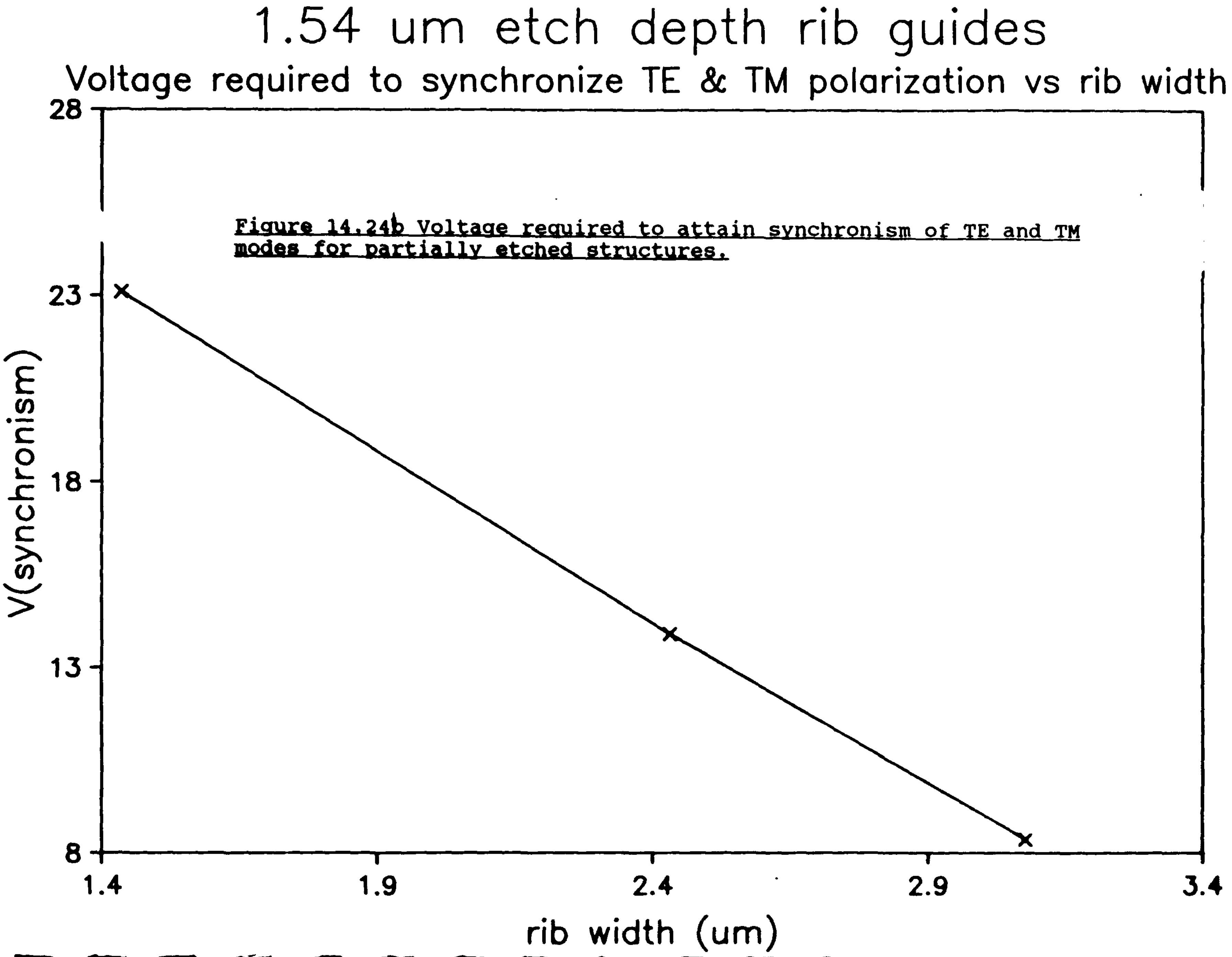


Figure 14.24b Voltage required to attain synchronism of TE and TM modes for partially etched structures.

high density arrays of a select few modulators, based on the results of the discrete devices.

The mask set for single devices encompasses waveguide widths of 1.4 to 3.0 microns in 0.2 micron increments for the single mode sections of the device, and double these values for the two moded sections (for example, at the splitting and recombining junctions of the Mach-Zehnder). While simply doubling the width does not universally guarantee two-moded operation, modeling indicates that the correct operation will be obtained for the structures incorporated in the mask set. The mask set makes provision for the use of both semi-insulating GaAs substrates and also n^+ material. While initial calculations indicate that the semi-insulating material will enable lower crosstalk to be obtained for modulator arrays compared to devices grown on n^+ material and using backside contacts, the techniques proposed by Walker (77) may enable comparable crosstalk to be obtained with higher drive frequency. Thus it was considered beneficial to incorporate the facility for the use of n^+ substrates.

The function of the various levels of the mask set and the process steps in the fabrication of the modulators are shown in figure 14.25. We describe the function of the mask levels by reference to a phase modulator structure. However, the process is very similar for more complex electrode configurations needed for the Mach-Zehnder.

A reactive ion-etch process is used to define the waveguide ribs, in conjunction with the first level of the mask set. Different etch times will be used for the fabrication of lateral contact polarization modulators. All other devices will use a common process.

The second level of the mask set is used to define, via lift-off, the contacts which will reside on the planar etched surface of the substrate.

In the third step, the substrate is spin coated with polyimide to planarize the surface. The need to obtain a uniform rather than a conformal layer results in a thicker film than desired for alignment. Thus a uniform oxygen reactive ion etch is used to reduce the thickness in the fourth step.

A third mask level is used to define both the polarizer top surface area to be exposed, and to reduce the polyimide thickness in areas where contacts to the metals previously defined on the substrate are to be made. In step 6, an oxygen reactive ion etch is used to reduce the polyimide thickness in the defined areas, resulting in the top layer of the polyimide being exposed. In the seventh step, a wet chemical etch is used to reduce the thickness of the AlGaAs top cladding.

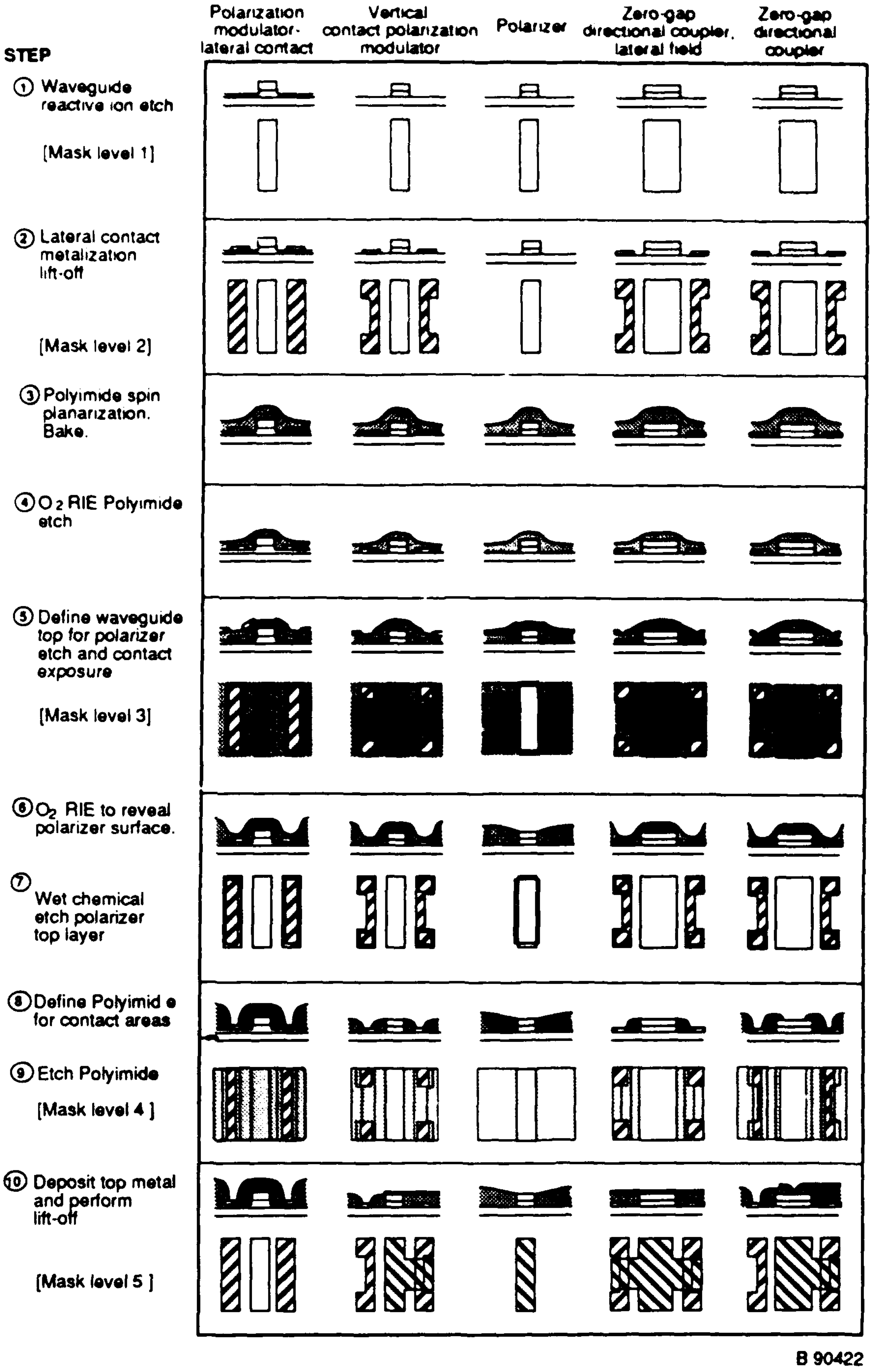

STEP
Polarization modulator-lateral contact
Vertical contact polarization modulator
Polarizer
Zero-gap directional coupler, lateral field
Zero-gap directional coupler
① Waveguide reactive ion etch
[Mask level 1]
② Lateral contact metalization lift-off
[Mask level 2]
③ Polyimide spin planarization. Bake.
④ O 2 RIE Polyimide etch
⑤ Define waveguide top for polarizer etch and contact exposure
[Mask level 3]
⑥ O2 RIE to reveal polarizer surface.
⑦ Wet chemical etch polarizer top layer
⑧ Define Polyimid e for contact areas
⑨ Etch Polyimide
[Mask level 4]
⑩ Deposit top metal and perform lift-off
[Mask level 5]
B 90422

Figure 14.25 Steps in the fabrication of AlGaAs waveguide modulators.

The contact areas are opened up by means of a further oxygen reactive ion etch, a fourth mask level being used to provide definition.

Next, the top contact metal is deposited and patterned using lift-off. The fifth and final mask level is used for this stage. The process as described will result in the same metal being used for the modulator contacts and the polarizer metal overlay. A further level would be required to enable different metals to be used.

In the case of array devices, yet another polyimide deposition step is used, in conjunction with a further mask. The purpose of this additional procedure is to add bridges between the lower polyimide and the bridging metal wherever metal lines are to pass over a device to provide contact to another. This reduces the probability of short circuits, which may occur if the thin (0.2 microns) of polyimide over the waveguide were used without enhancement.

The Mach Zehnder modulator is not shown in the figures, but has a structure essentially the same as the vertical contact phase modulator. In plan view however, the device is seen to incorporate waveguide junctions at the input and outputs. Critical to the low-loss operation of both the two-mode interference based devices, and the Mach-Zehnder interferometer, are sharp waveguide intersections. The design of the Mach-Zehnder interferometer was described in the last quarterly report. The designs incorporated into this mask set have gradual waveguide bend, consisting of straight segments intersecting at 0.1 degrees, with the exception of the initial waveguide split, which has a **total** angle of 0.5 degrees. The straight waveguide sections have a length of approximately 100um. No attempt has been made to optimize the section length with regard to minimisation of bend loss: careful engineering may later be employed to reduce the total insertion loss via the use of coherent bends (76).

Critical to the realization of low-loss devices will be the realization of highly sharp apexes to the waveguide splitters. Extinction ratio in Mach-Zehnders depends largely on symmetry, and therefore on careful processing. Excess loss however depends of obtaining sharp splitters. Process development is aimed partially at the definition of such a sharp apex and its maintenance throughout the process.

Provision was also incorporated into the mask set for fabrication of guided wave polarizers. In order to be integrable with the polarization rotators, selective areas of the waveguide are reduced in thickness, after which they are coated with dielectric and metal, or just metal. This involves an additional level of masks incorporating the etch windows. This relies on a modified growth procedure being used, specifically the incorporation of a thin layer of AlAs at a predetermined point in the upper cladding layer.

14.4 Materials Growth

Our aim in growirq the materials for the modulators is to produce low carrier concentration material. High carrier concentrAtion will have two adverse effects. First, free-carrier absorption will contribute to the total optical insertion loss of the device. Second, as the carrier concentration increases, a greater voltage will be required to fully deplete the waveguide region, hence a greater voltage must be applied to the device to achieve a given field within the waveguide region. Since the breakdown voltage of a practical device is of the order of tens of volts, effective operation of the device may not be possible. Note that breakdown will only result in damage to the device if the associated current is not limited.

The loss contributed by free carrier absorption may be calculated from [78]. At a concentration of 10^{16} the attenuation would be .03dB/cm, while at a concentration of $10^{17}/cm^3$ the attenuation would be .3dB/cm. These attenuations represent the loss with no applied voltage. As the carriers are depleted, the absorption will decrease, until for fully depleted guides no free carrier absorption will be present.

The structure grown for all experiments with the exception of the polarization analyzers, consisted of an AlAs superlattice buffer on semi-insulating GaAs. On the superlattice buffer are grown 3 microns of 35% AlAs composition AlGaAs, followed by one micron of 30% AlAs concentration, and a further micron of 35% AlAs concentration. This structure was common to all the experiments in the discrete device development phase.

Some of the materials for the modulators were grown n three inch wafers. Uniformity over such an area with our existing reactor design is approximately 15%. This deviation arises from a desire to keep the thickness controlled in the center of the wafer, while relaxing the constraints away from the center. Carrier concentrations ranged from 7 x 10^{14} to 1.5 x $10^{16}/cm^3$. The measurements were obtained using both C-V and Hall effect procedures. Growth was performed on both semi-insulating and on n^+ substrates, while maintaining good surface morphology.

14.5 Processing

During early processing experiments, it was apparent that several defects were present in the fabricated waveguides. These were (1) striations in the waveguide walls, and (2) a step, or series of steps, in the waveguide walls (ie a non-vertical boundary). The first defect would result in increased waveguide wall scattering, and hence a reduction in throughput, while the second will contribute to loss via variation in the height of the step with propagation distance. This would result in lower performance than ultimately possible, and in variations between process runs, precluding the refinement of the modulator processing.

Striations in the waveguide wall are caused by transfer of imperfections in the contact mask used into the resist employed to pattern the waveguide. The imperfections arise from the etching of the chrome, and are essentially identical to those encountered in the 10x masks used for projection lithography. In the case of projection lithography however, the imperfections are smeared out in the reduction process, ultimately resulting in the smoother sidewalls typically obtained for this process.

The steps in the sidewalls were believed to be caused by a non-vertical wall in the photoresist mask used to define the waveguide, and by lateral etching of the mask during processing. The step is common to both contact printing and projection lithography, thus no improvement would be obtained by the use of the three inch process. As the resist is etched down during the waveguide etch process, the non-vertical wall manifests itself as a change in the masked area above the rib. The presence of the step is not as significant as the variation in its height along the length of the device. This potentially results in a variation in the waveguide cross-section with propagation distance, and hence some loss due to scattering. The desire to eliminate variables between runs suggests that the step should be eliminated, even if its contribution to device loss should be determined to be negligible.

Initial efforts were directed at either reducing the width reduction associated with the ridge via the use of a silicon nitride layer between the resist and the upper surface of the top AlGaAs layer, or in displacing it to the cap layer above the waveguide. It was hoped that the use of silicon nitride would either prevent or eliminate the reduction in width of the masking layer. Unfortunately, the etch parameters required to fabricate the AlGaAs rib waveguides at an acceptable rate appear also cause etching of the silicon nitride. The nitride served merely to reduce the magnitude of the defect, rather than to eliminate it, thus this approach was not adopted.

The solution found to be most effective involves the use of an SiO_2 mask layer directly above the grown AlGaAs layers.

In order to reduce the magnitude of the striations, and hence the scattering loss of the waveguide several experiments were performed. During the spinning of photoresist onto the sample, a bead is formed around the edge of the substrate. An experiment was performed to determine whether the presence of this bead, with consequent physical separation of the mask from the resist, was contributing to the striations. In order to investigate this hypothesis, two samples were prepared. In one, the resist was spun as usual, while in the second the bead was removed from the edges. Processing for the waveguide delineation then proceeded as usual.

The results of this experiment were inconclusive, with only a small difference being observed between the devices with and without the bead removal. This led us to believe that the defects on the mask would always be transferred onto the resist. We therefore sought a solution which allowed the defects present to be reduced.

The process finally used is not susceptible to these defects, and involves depositing 7000A of SiO2 on the AlGaAs surface, on the surface of which is spun photoresist in the normal manner. Reflowing of the resist at elevated temperature is performed to smooth out the local perturbations in the patterned edge of the waveguide. Reactive ion etching then proceeds as normal. The SiO_2 has been shown to resist erosion in the etch, and thus maintain vertical sidewalls. We note that the imperfections are originally caused by the contact process. Some reduction in their magnitude would be expected from the use of projection lithography. The experiments were however designed to use smaller pieces than the three inch wafers required by a projection process, in order to maximize the amount of information obtained from the experiments.

Figure 14.26 shows the waveguides at various stages of the processing. Slight rounding of the lower corners of the structure will be observed. This was not intentional, but a result of the process once optimised to reduce sidewall scattering.

Figure 14.27 illustrates the active structures by showing part of an array of 10 polarization rotation devices. The linear array of bond pads can clearly be seen, as can the bridges across the waveguides to access the more distant modulators.

14.6 Waveguide and modulator testing

In order to characterize the device performance, several key parameters must be determined. These are

1) Device insertion loss and propagation loss

2) Drive voltage required to effect a certain change in the properties of the light propagating through the device, for example the voltage required to achieve a minimum in transmission of a device with a periodic transfer function, and

3) The maximum achievable extinction ratio, or other such parameter.

Modeling had earlier established that synchronism of the two orthogonally polarized eigenmodes of the polarization rotator would limit the attainable extinction ratio. The three-electrode modulator is, in principle able to overcome some of these deficiencies, however for a given applied voltage, compensation may only be provided for a finite synchronism. The maximum tolerable

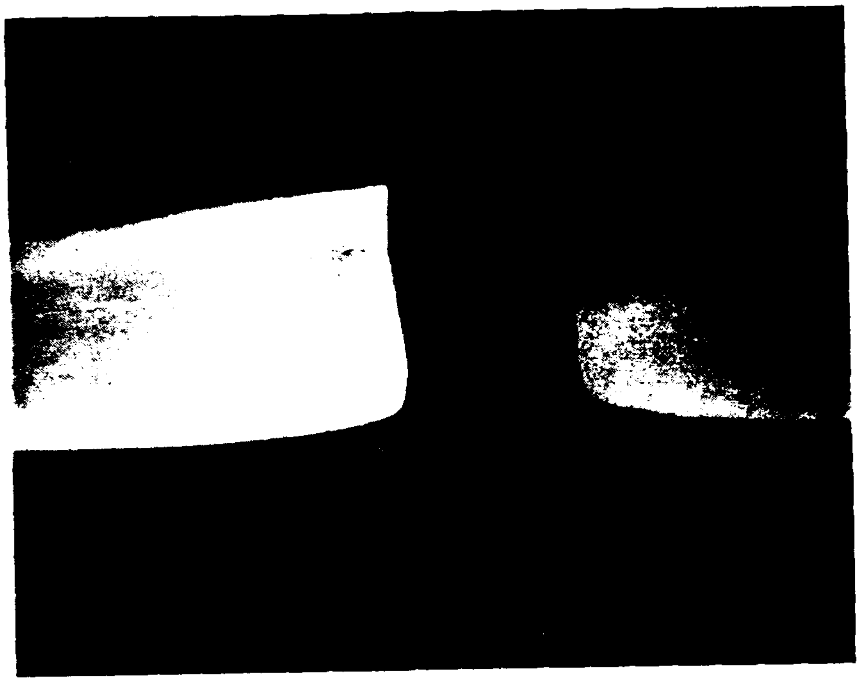
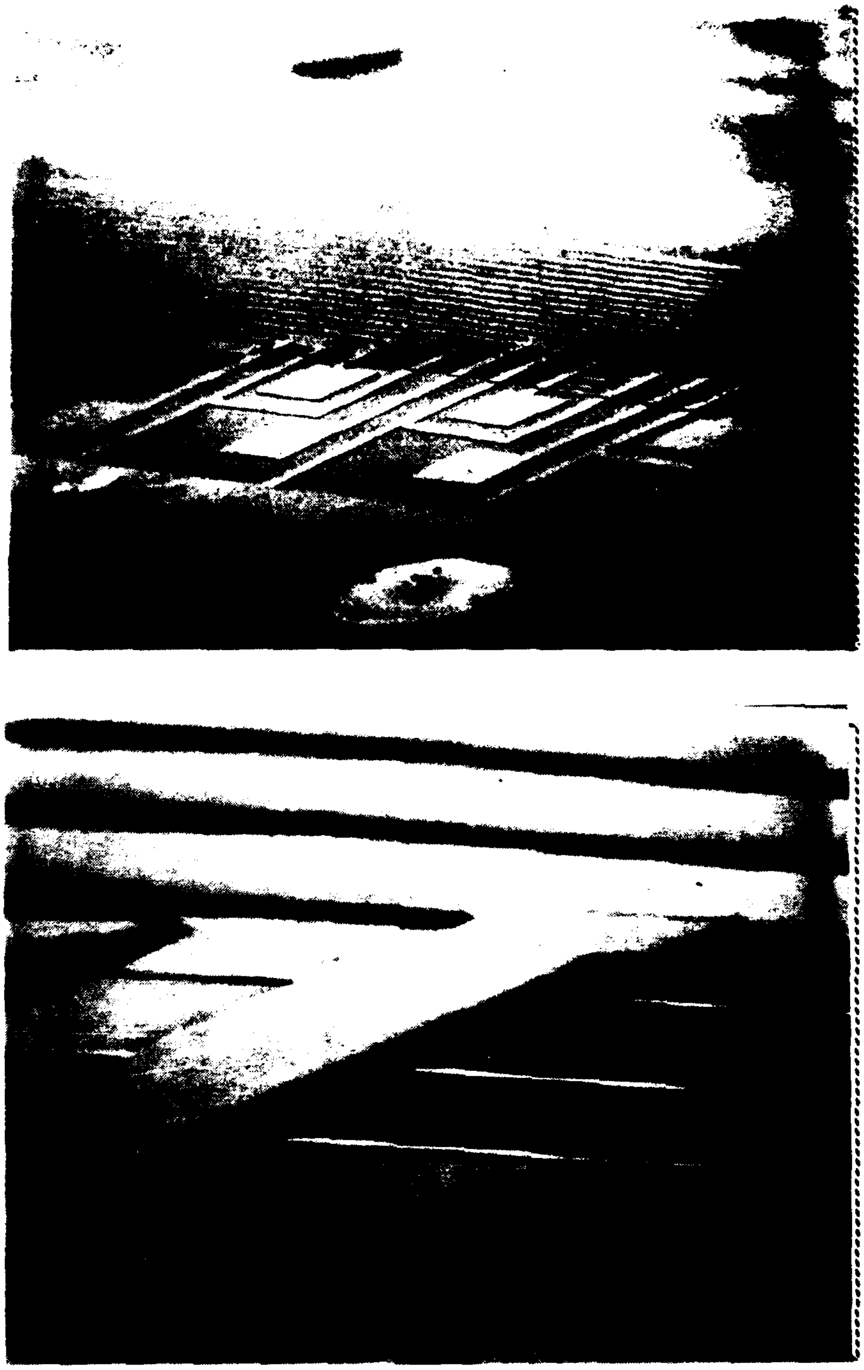


Figure 14.26. Illustrating waveguides after RIE using reflowed resist. Waveguide width is approximately 1.8 microns.



B00730

Figure 14.27. Array of 10 polarization rotation modulators, and the polyimide bridges used to interconnect devices and bond pads.

asynchronism is thus determined by compatibility with control electronics, or ultimately by the breakdown voltage of the fabricated device. Thus an important part of the experiments involved determining the variation in maximum attainable extinction ratio with control voltage.

14.6.1 Waveguide performance

The ratio of TE to TM attenuations (ie. the differential loss) was determined as shown in figure 14.28 by rotating the incident plane of polarization using a Soleil-Babinet Compensator set up to function as a half-wave plate. The adjustment of the compensator is performed between crossed polarizers in the absence of and lenses, thus ensuring that lens depolarization does not corrupt the accuracy of the initial adjustment. Figure 14.29 shows the ratio of TE to TM outputs for equal excitations in the case of a basic waveguide with no metal overlay, and the case where Ti/Au is deposited on top of the one micron buffer layer. As had been expected, the ratio of the transmissions was almost unity for the unclad guide. The addition of the metal to the upper cladding induced a factor of two difference in transmissions for the two modes. Also, the differential loss appeared to increase abruptly around 2.5 microns, this corresponding to the onset of the second order mode, with presumably less confinement than the corresponding fundamental mode.

Figure 14.30 shows the ratio of transmitted power for unclad and metal clad guides, the measurements being performed for both TE and TM excitations for a range of waveguide widths. The device length in this experiment, as in the last, was 9.3mm. The additional loss of approximately 1dB/cm associated with the TM modes is clearly seen.

14.6.2 Phase modulator performance

Before evaluating the performance of the more sophisticated modulators, a basic phase modulator was investigated. Figure 14.28 shows the experimental arrangement used to test the basic phase modulator performance of the active structures. Rather than set up an external interferometer to determine the phase modulation, the input plane of polarization was oriented at 45 degrees to the surface of the device. The phase modulation properties were determined on the assumption that only the TE component of the electric field was affected by the applied field. An analyzer oriented at 90 degrees to the input polarizer converts the polarization rotation to an amplitude modulation.

The best extinction ratio determined for this category of device was 20dB, this being for sample 1390B, which was fabricated on semi insulating material. The carrier concentration was 1.4 x

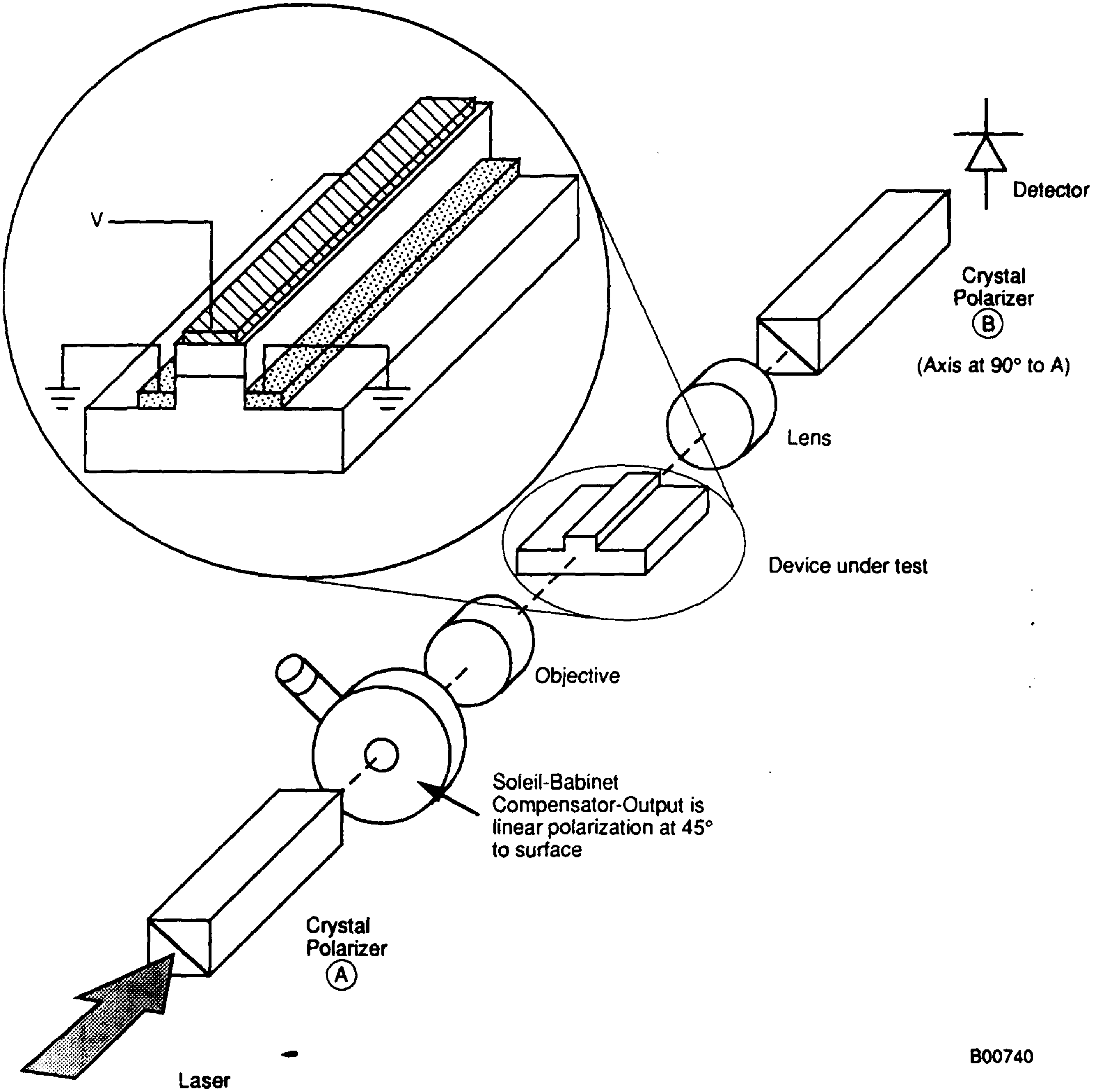


Figure 14.28 Test arrangement for phase modulators

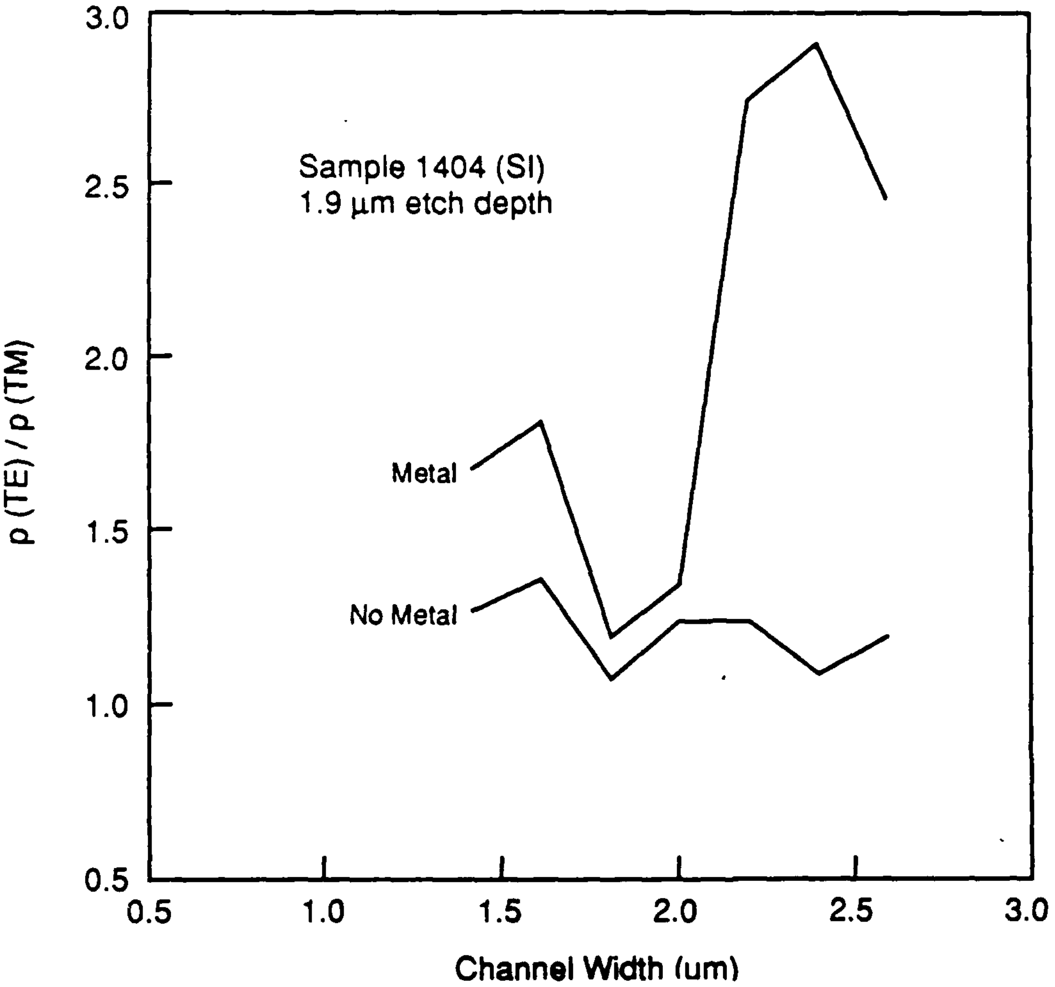


Figure 14.29 TE/TM transmission ratio for 9.3mm sample length

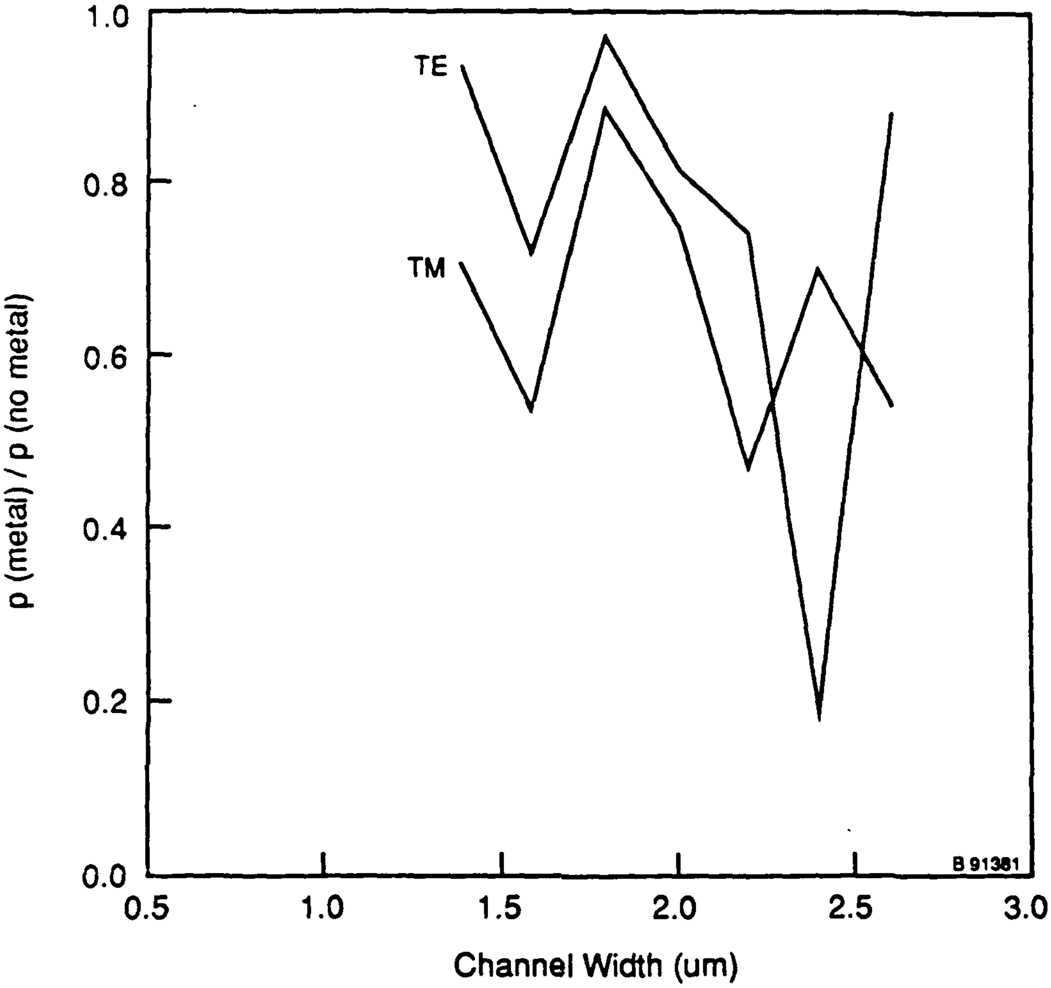


Figure 14.30 Ratio of transmissions for metal clad and unclad guides of 9.3mm length.

1016 (p-type). The etch depth on this sample was 1.9 microns, slightly less than the value od 2.0 microns which would correspond to complete etching of the guide layer. The nominal waveguide width was 2.2 microns. The maximum extinction ratio was obtained for a total applied vcltage of 12 volts, for a device length of approximately 8mm. While the transition from undepleted to fully depleted operation is not a step function, no interaction between the applied field and the optical transmission was observed until 5 volts had been applied. Also, due to the desire to save the device for further experiments, is was not deemed desirabale to increase the voltage by the further 14 volts to obtain a second maximum with fully depleted operation. The propagation loss of these waveguides was typically 1dB/cm, somewhat higher than the values of 0.2dB which had previously been obtained from this structure. The addition of the metal to the basic waveguide results in an additional loss of approximately 1dB/cm for this width. Figures 14.29 and 14.30 shows the differential transmission of clad and unclad guides as a function of width. The wavelength at which these experiments, and all others reported here, were undertaken, was 820nm.

14.6.3 Mach-Zehnder Performance

Figure 14.31 shows the experimental arrangement used to test the Mach-Zehnder interferometric modulators. For convenience, and because the breakdown voltage of the modulators was typically in excess of 30 volts, the modulator was used in a single-sided configuration. This may affect the ultimately attainable extinction ratio, since the other waveguide will contain some carriers, while the modulated guide will have them swept out. Thus the absorption loss due to free carriers will be asymmetric.

The input and output polarizers shown in the figure were used to ensure that only TE inputs were used, and that depolarization through the length of the waveguide modulator did not degrade the extinction ratio measurements.

The active electrode length for the Mach Zehnders was 5mm, while the total length was just under 1cm. This allowed for a significant portion of input and output single mode guides at either end of the device to avoid the corrupting influence of the propagation of the higher order modes in a leaky manner, which would occur if shorter lengths were used.

Figure 14.32 shows a graph of transmission versus applied voltage for device 1399A, this having a carrier concentration of 7×10^{14}. For a pure TE input, outputs in both TE and TM states are shown. Interestingly, the finite TM output shows a lower voltage required for extinction than the TE output. The latter is of course the mode of interest for this device. Full extinction was measured to be 23dB for a width of 2.2 microns (nominal). The voltage required to obtain this value was 22.4 volts. It is seen that little effect on the transmission is observed until the applied voltage has exceeded 13 volts. Presumably after this, the edge of the depleted region begins to intrude into the waveguide, and some modification of the index results. The departure of the curve from a perfect cosine function would confirm this behavior. Assuming then that 11.5 volts corresponds to a pi phase change for the 5mm active part of the structure, we find that V_{pi} would be approximately 55 volt.mm. This rather high value can be considered to be a result of the thick upper cladding, designed to minimise the absorption of the TE mode due to the metal overlay. The excess loss for the complete Mach-Zehnder, over and above the loss of a similar straight waveguide, was approximately 2dB.

We note that this extinction ratio represents the highest yet obtained for a Mach-Zehnder fabricated on AlGaAs/GaAs and operating at 820nm wavelength.

14.6.4 Polarization modulator performance

Earlier, the design of a polarization based modulator was described. In order to achieve phase-matching of two orthogonal eigenmodes, a modification to the conventionally encountered two electrode polarization modulator was introduced. The device is illustrated in figure 14.32. The material from which the device was fabricated, 1449, had a carrier concentration of 3.6x 10^{15}. Three electrodes are employed with the aim of providing two independent fields, one horizontal and the other vertical. This will result in phase-matching for the modes corresponding to polarization states at 45 degrees to the waveguide wall and top surface.

It will be observed that the electrode configuration for this device is essentially the same as for the phase-modulator type structures incorporated in the Mach-Zehnders. The differences between the two lie in the etch depth of the structure and in the ability to separate the potentials of the two lower electrodes. The device functions by applying a vertical field to the waveguide region, with the effect of inducing a change in the effective index for the TE mode. A lateral field is also imposed, which results in new principal axes being defined at 45 degrees to the waveguide upper surface. Thus a TE or TM input signal is resolved into two mutually orthogonal components. The sign of the index change associated with each axis is opposite, so that a polarization rotation results. Thus with increasing application of voltage, a TE input would be converted to first TM, then TE again. Should any asynchronism be present in the waveguide, the efficiency of the conversion will decrease in a sin(x)/x manner.

The device was tested by applying a fixed potential to the upper electrode, and then recording the voltage required to minimize the polarization component parallel to the original. Both the required voltage and minimum transmission between crossed

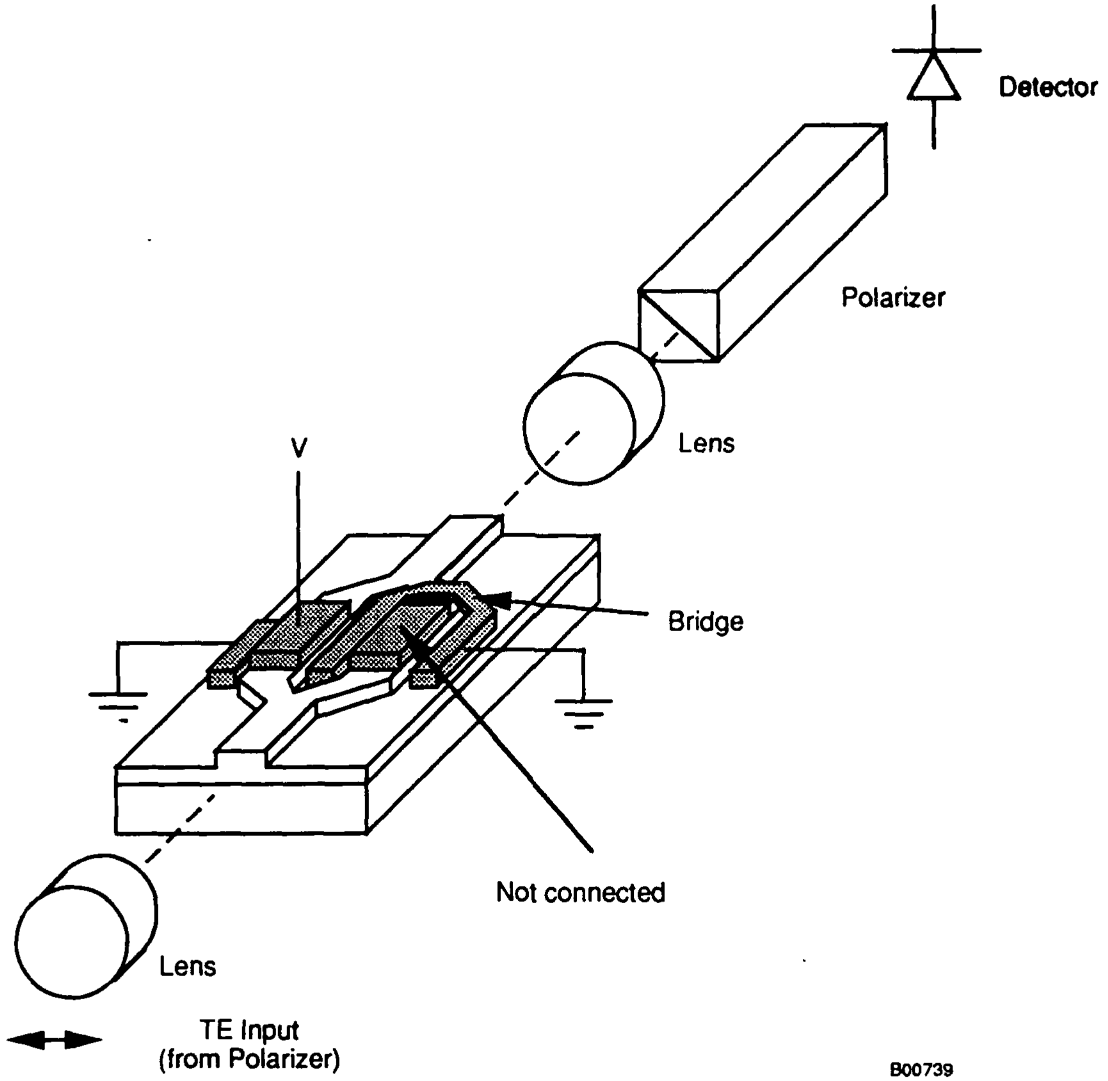
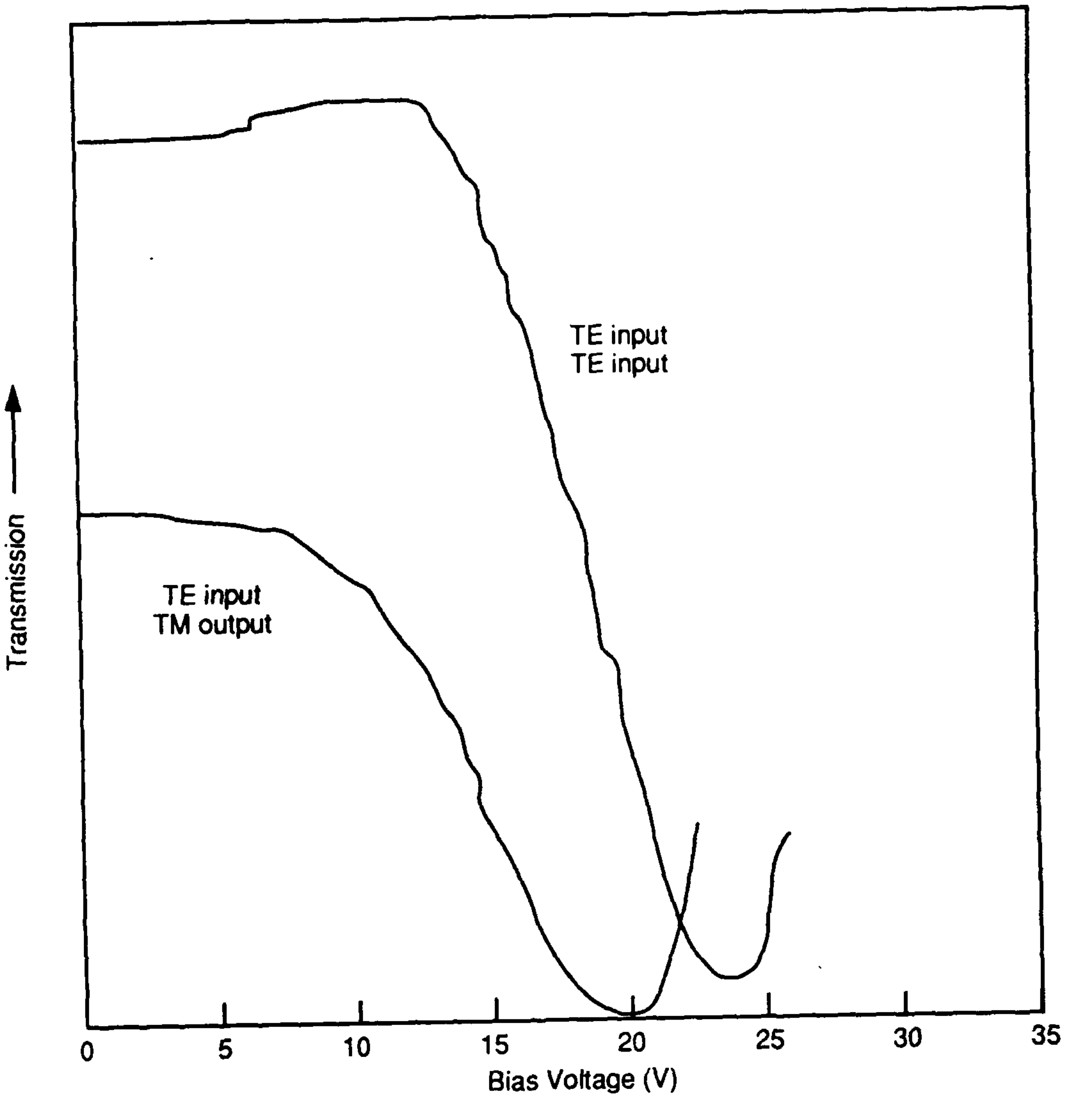


Figure 14.31   Experimental arrangement for testing Mach-Zehnders



B00735

Figure 14.32 Mach-Zehnder Transmission.

polarizers were recorded. Figure 14.33 shows the variation of the transmission as a function of lateral field for several different values of the applied vertical field. It will be noted that the voltage required to minimize the extinction actually increases with increasing bias. These results can be explained as follows. A given applied vertical field results in either an increase or a decrease in the effective refractive index of the TE mode in this structure. When fabricated, some birefringence will be built into the guide. In other words, the TE effective index may be smaller than or larger than the TM index by some amount. If the carrier concentration in the active region of the device were negligible, it would be a simple matter to select the appropriate polarity of the field to offset the birefringence. Unfortunately, appreciable carrier concentration in this p-type material results in depletion being localized in the region of the negatively biased electrode. If this electrode happens to be that above the rib, most of the applied field will appear in the waveguide region. If however the birefringence requires that this electrode be positively biased, so that the negative bias is applied to the more distant surface electrode, a much greater voltage will be required to offset the birefringence, and may in fact cause breakdown before phase matching is achieved.

In figure 14.34 we show corresponding results for the direction of propagation orthogonal to that of the previous device.. Now the correct mode of operation can be seen, wherin increasing the vertical potential acheives better synchronism of the orthogonally polarized eigenmodes. The vertical and horizontal fields are obviously not completely decoupled, since the voltage required to achieve rotation increases with applied vertical field, contrary to the theory predicted by decoupled fields.

14.6.5 Anomalous polarization behavior

During the evaluation of the fabricated samples, it was noted that even in the absence of an applied field, a linear input polarization parallel with the waveguide axes resulted in an elliptical output state. This implies that some depolarization and polarization rotation mechanism exists within the device. Possible causes include stress in the waveguide region. Great care was taken to ensure that the input and output microscope objectives did not contribute to the depolarization. In particular, strain-free objectives were employed. These maintained a linear state of polarization with a depolarization better than -40dB. The depolarization and polarization rotation are thus attributable to the waveguide device, even though the exact mechanism responsible for the effect is not known.

14.6.6 Polarization Analyzer Behavior

The polarizers fabricatea by the selective etch process detailed in the fabrication description were tested for differential transmission as described earlier. During testing, it became apparent that the cladding parameters were not optimised. The maximum extinction ratio obtained was 22dB, but this was at the expense of a 10dB excess loss. These results suggest that use of the index change due to the fiffering AlAs concentrations in a multilayer structure will not result in a high performance device. Further experiments would use a separately deposited dielectric of lower index to couple the guided TM mode to the surface plasma wave.

14.6.7 High Frequency and Array operation

Arrays of devices were incorporated in the second mask set. Groups of 5 Mach-Zehnders and four polarization rotators with different waveguide widths. Widths of 2.2 microns were used to construct arrays of 10 of each type of device. The spacings of the polarization rotators were 25 microns and 100 microns. The polarization rotators were equipped with isolation grounds between adjacent pairs of devices, as shown in figure 14.35. Since the mach-Zehnders have all lower metal at ground, introducing additional grounds would serve no purpose.

The array performance was evaluated by testing each device in turn. The highest yield was obtained for the array of polarization rotators, eight of which were determined to operate correctly. The remaining two in the array of 10 were found to suffer from shorts.

The high frequency performance was evaluated using the arrangement shown in figure 14.36. A 50 ohm termination is provided for the RF source, via a bias tee to enable the depletion voltage to be applied. The value selected for this was half that needed for complete extinction, to maximize the small scale sensitivity of the modulator. Figure 14.37 illustrates the non symmetric nature of the transfer function of the device, due to incomplete sweeping out of the carriers within the waveguide.

The frequency response of the device was determined up to 1GHz, beyond which the response of the probes was so poor that accurate deconvolution was not possible. The devices had not rolled off by 3dB when this upper frequency limit was reached.

Crosstalk between adjacent devices was measured for the Mach-Zehnders. In order to simulate as accurately as possible a practical case, the adjacent device was biased to an identical operating point. The modulation of light travelling through this adjacent device was then determined, with a signal applied to the original device. The crosstalk was found to depend on frequency.

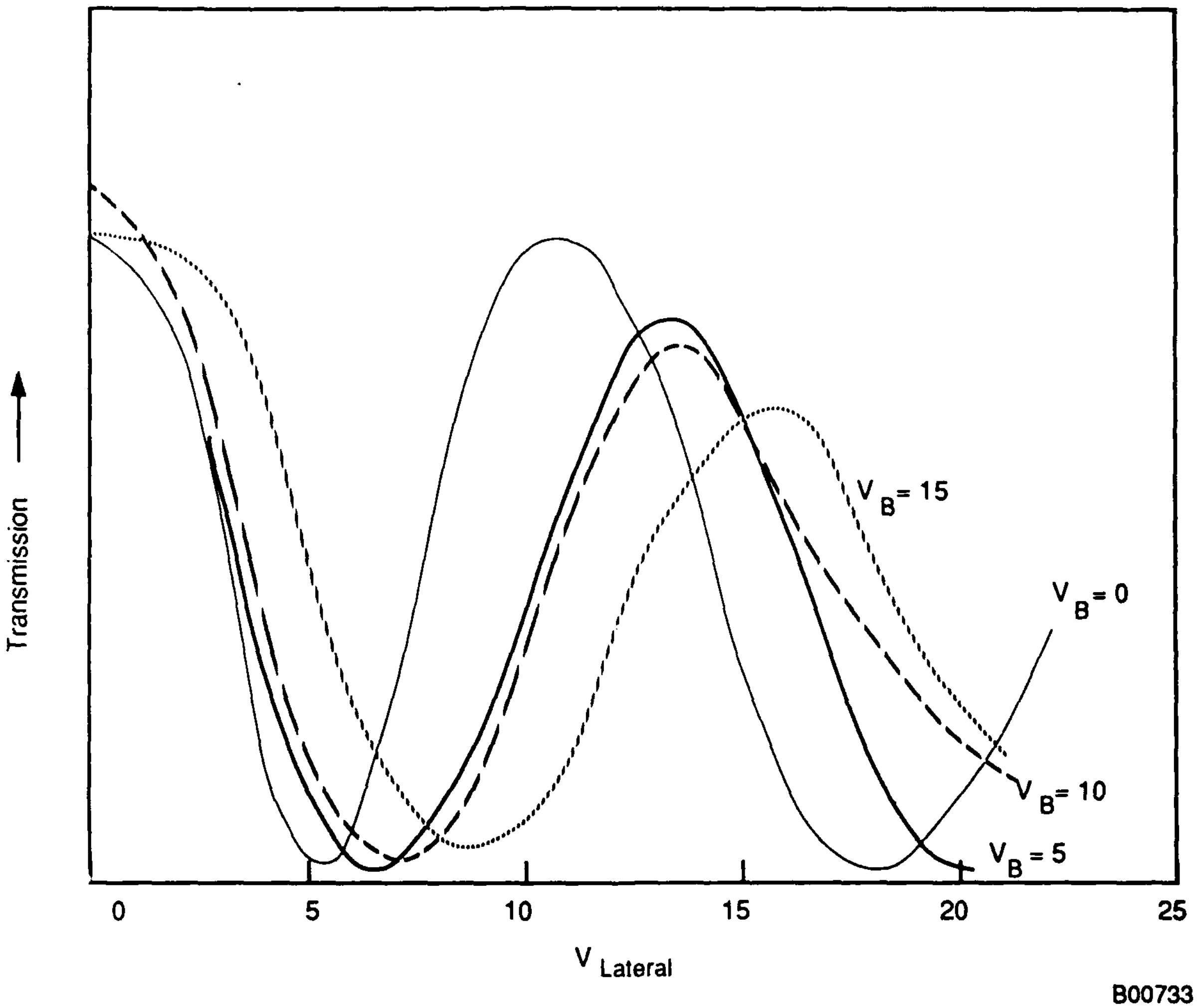
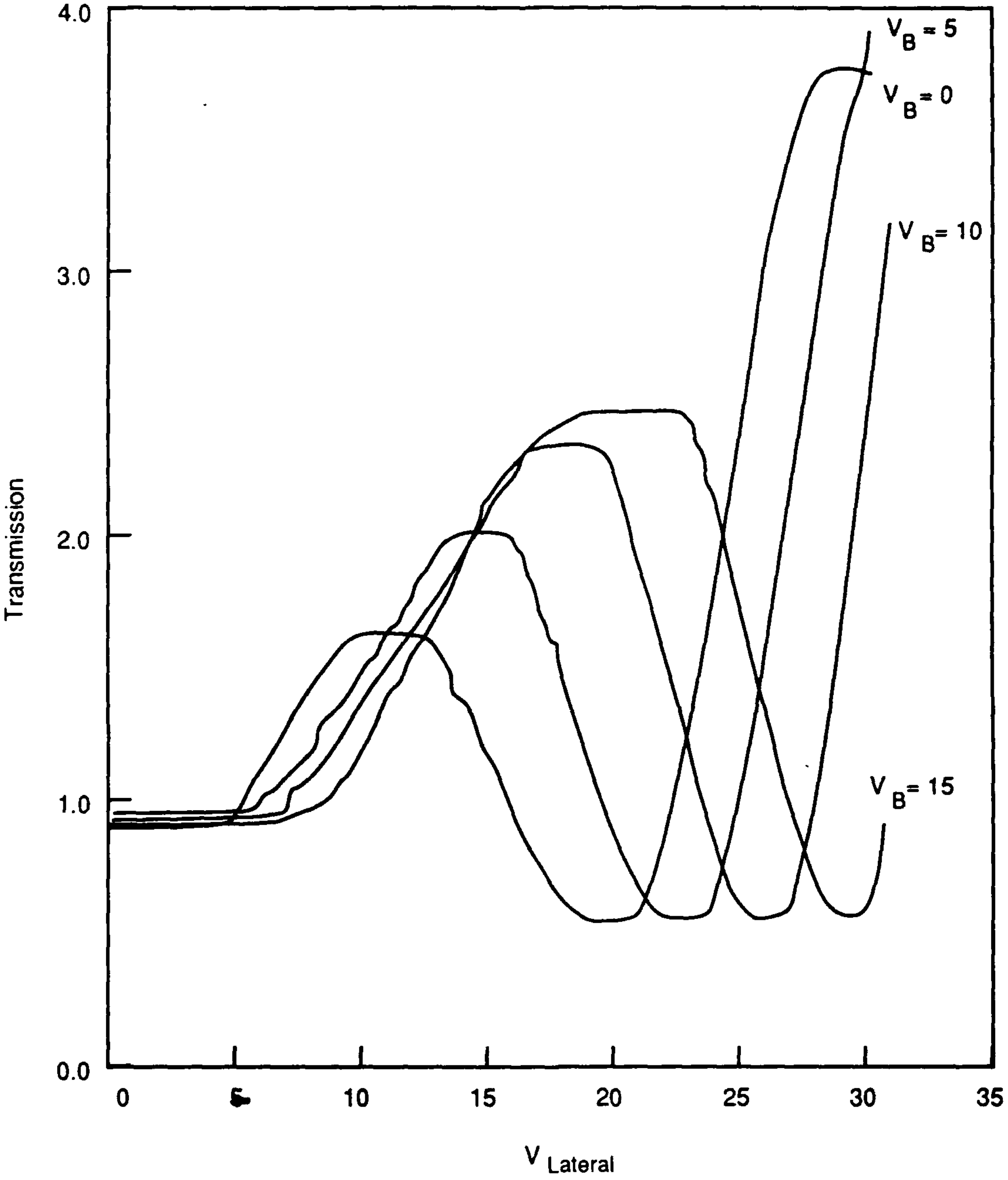


Figure 14.33 Measured throughput of polarization rotator as a function of lateral voltage for several different bias voltages.



B00732

Figure 14.34 Measured throughput of polarization rotator as a function of lateral voltage for several different bias voltages, but oriented at 90° degrees.

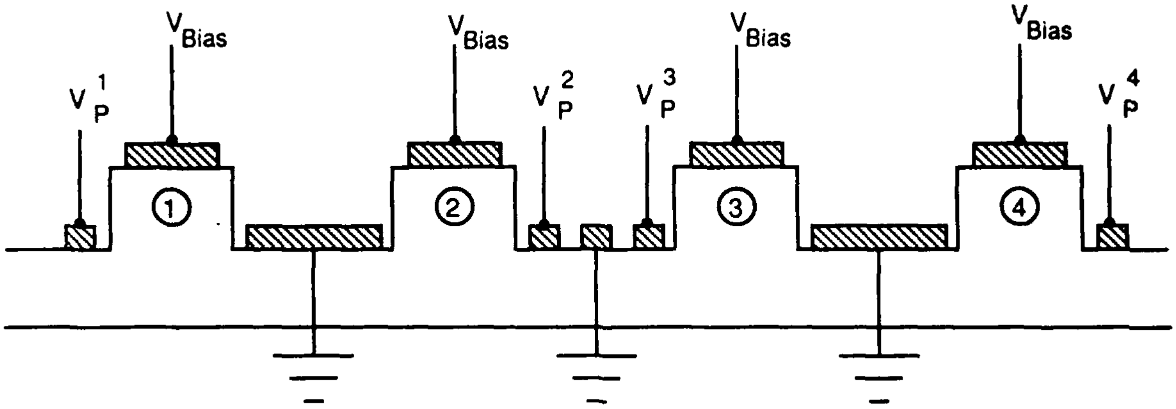


Figure 14.35. Grounding arrangements for polarization rotation arrays.

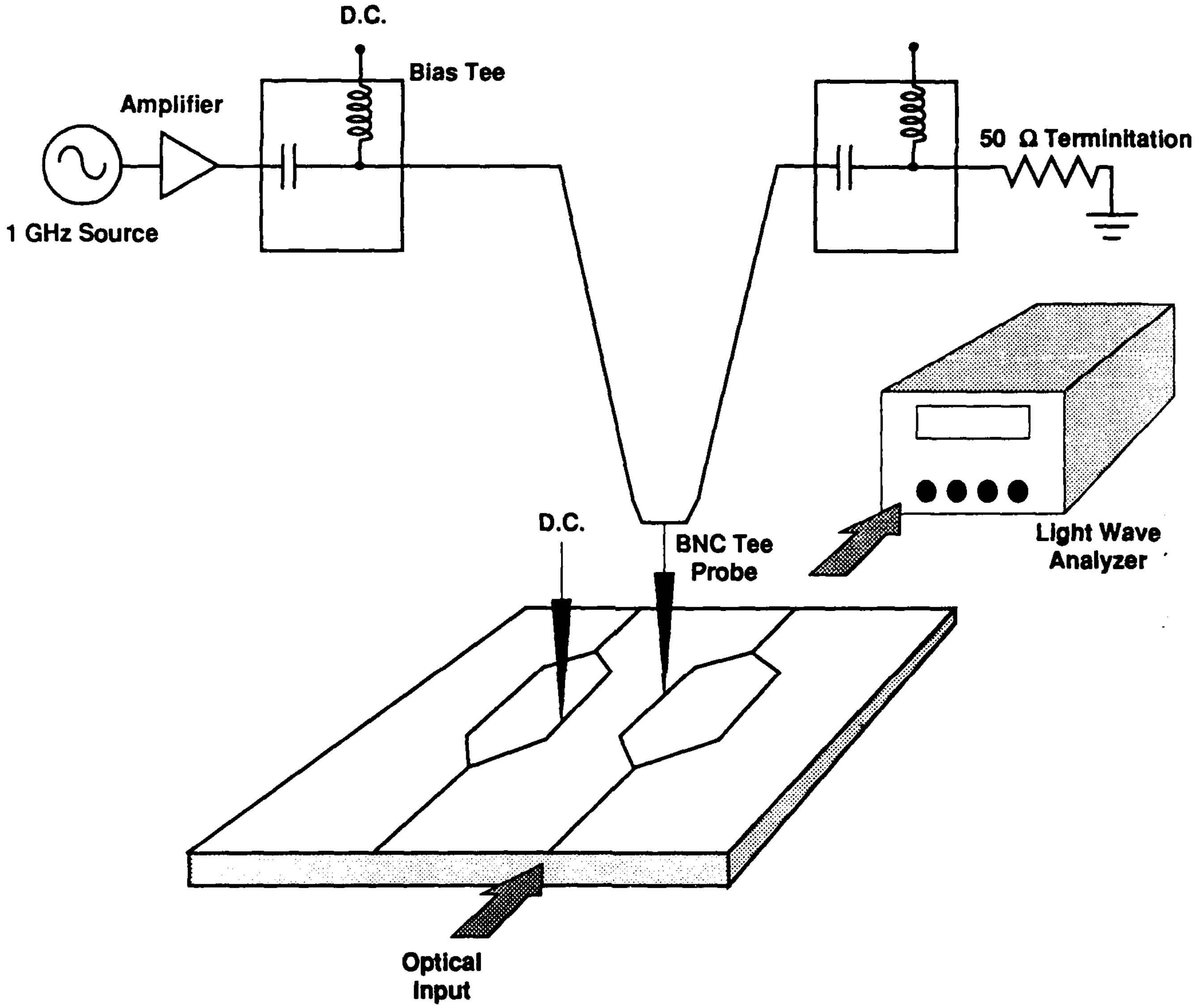


Figure 14.36 Experemental arrangement for determining crosstalk and high frequency performance (DC bias and the two tees are not used for crosstalk determination)

B00738

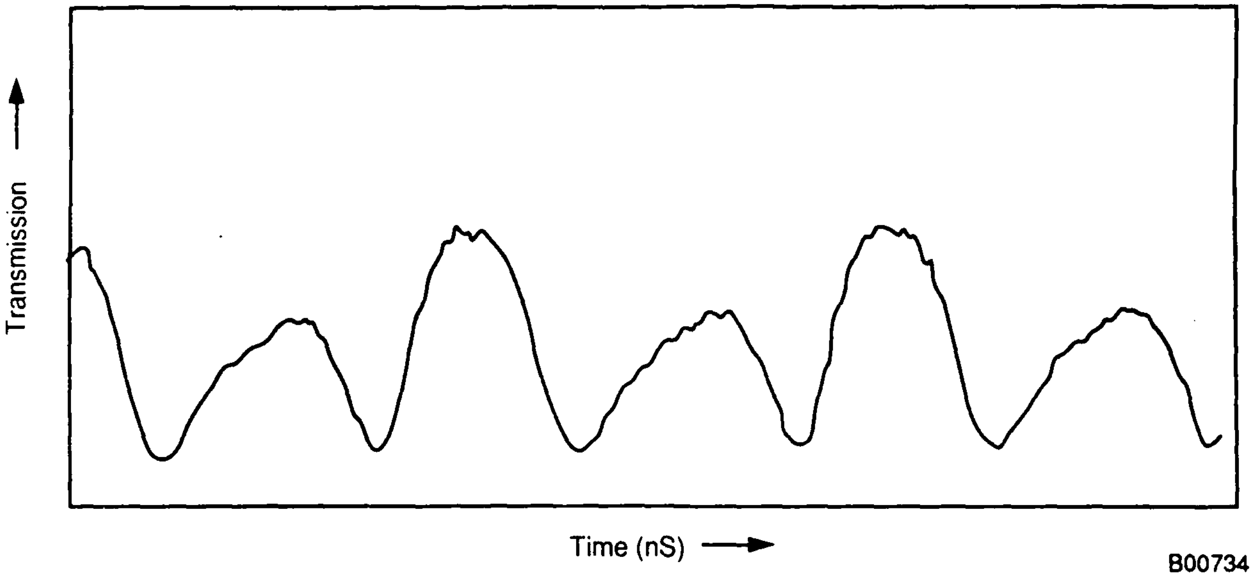


Figure 16.37. Asymetric transmission junction of Mach-Zehnders for Sinusoidal drive voltage.

At a frequency of 290 MHz, the (electrical) crosstalk was -22dB. While insufficient data are available to determine accurately the variation in crosstalk with frequency, it was noted that the value was significantly worse at approximately 60MHz, and 130MHz. The frequency difference between these values corresponds to a wavelength of some 5 metres. This is unlikely to be associated with any property of the modulator array, rather the long leads used to connect the probes to the sources and bias power supplies. Similarly, little improvement in crosstalk was observed when the second, third and fourth most remote devices were accessed with the RF probe, indicating that lead pickup may be the dominant crosstalk mechanism. The quoted vales of -22dB are perhaps therefore conservative.

15. Conclusions

The goal of this program was to develop key concepts for optical symbolic computing. The goal was accomplished using both a top-down and bottom-up approach.

The approach initially taken was to examine computational models of computer languages, determine primitive operations required, and develop and evaluate a conceptual architecture. It was found that the computational requirements of logic languages and functional languages are primitive operations which involve manipulation of complex structures such as graphs and trees, and that the execution of the languages can be described as manipulations of those data structures. The representation of the complex data structures imply that the representations must be exact, and therefore for all practical purposes digital, and that some means to denote connections between the data items, such as pointers, is required. Since the representation between data items is more important that the actual items stored, the most important functions are the manipulations of the data structures.

Examinations of the optical architectures available to represent and implement the identified functions showed that some way to perform location addressable memory was needed. One technique, matrix representation, was identified and a technique to construct addressable optical memories was invented. By examination of a possible architecture however, it was found that these methods do not adequately perform the computational primitives. Moreover, it was found that while functional languages and logic languages require similar primitive operations, implementation of logic languages in parallel optical environments is more difficult.

Symbolic substitution was evaluated for optical computing. It was found that symbolic substitution by itself cannot perform the required memory and data movement functions, but is well suited to the control functions required in a computer. We examined the possibility of combining other optical computing structures with symbolic substitution to perform the data movement and storage to develop a viable computer architecture.

A novel architecture for an optical symbolic computer was developed. The architecture, Symbolic Processing Architecture in Optics, is designed for executing functional language programs using combinator graph reduction. Sparo was designed with the goal of exploiting the available fine-grained parallelism of both combinator graph reduction and primitive optical operations. A planar array of processors communicating by messages over a network provides the processing power of SPARO. The finite state machine of individual processors is expected to be implemented using symbolic substitution techniques, while gatable interconnects would be used for realizing data movements between the processor and the network. we proposed a simple register based network that would enable multiple messages to be delivered concurrently. It is shown that the architecture can easily be scaled to accommodate large combinator graphs. The detailed control sequence required for processing within the nodes and for messages are shown in as macro instructions that would be executed by each processor in SPARO. These macro instructions can be further translated into simpler symbolic substitution rules.

The optical implementation of the developed arhitecture was investigated. Since we had determined that the interconnection network was the bottleneck in the performance of the architecture, our focus was on the message throughput of the simple register-based network. We derived an accurate performance model for the equivalent bidirectional ring network and found, both by analysis and simulation, that the net parallelism in the architecture was restricted by the low message traffic in the network. When messages exhibited no locality, the throughput for a 1024 processor network was limited to 8. With local messages, the maximum throughput for the same network was 27.

The poor performance of the simple ring network motivated us to examine other more elaborate but efficient interconnection network topologies. The alternatives considered were hypercubes, multistage interconnection networks (MINs), and single-stage shuffle-exchange networks (SENs) and replicated SENs. On the basis of analysis and extensive simulations, we found SENs, especially replicated SENs, to be the most feasible and promising. Recent investigations have indicated that SENs could be implemented efficiently in optics. Furthermore, we established that replicated SENs can provide a high throughput competitive with any other interconnection network.

While the shuffle connection of the SEN is feasible in optics using passive devices, a full-scale exchange switch is not possible, due to the need to resolve conflicts in even lightly loaded networks. A reasonable alternative appeared to be to construct the basic exchange switch, and then incrementally add the necessary functionalities. We found that while the basic switch and the representation of the message can be done with relative ease in optics using different information encoding techniques, the conflict resolution function is far too complex to be implemented optically. Even using brute-force techniques such as holographic look-up tables to implement combinational logic that underlies the exchange switch, a large network (1024 or more) would require exchange switches of prohibitive sizes. We conclude that optically controlled network exchange switches will be a reality only when optics technology promises basic switching logic to be competitive in size and speed with electronics.

In conjunction with our work on the optical exchange switch, we also evaluated the advantages and the relative feasibility of hybrid optical designs for the complete SEN. We evaluated electronic and hybrid SEN implementations in terms of complexity and performance. The hybrid design refers to the use of an electronic exchange switch in conjunction with an optical shuffle connection. The analysis of electronic SENs required designing the interface between the processors and the SEN, the smart exchange switch, and means of laying out the perfect shuffle within the board. We considered both GaAs and ECL technologies to determine the highest performance of an electronic SEN. Our results showed that when a large number (1024 or more) of specialized graph reduction SPARO processors, whose complexity and sizes were estimated on paper, are packed on a board for high speed parallel computing, there is a severe performance degradation due to the limited parallelism in transferring messages. Our focus was therefore directed to using optics for implementing a high-bandwidth and high-density SEN multiboard architectures.

The requirement of high density I/O for boards is not unique to SENs. This was based on our analysis of the general I/O requirements of parallel architectures that are implemented as multiboard systems using other interconnection networks such as crossbars and hypercubes. A formal analysis of board I/O requirements in transferring messages in parallel between PEs was conducted to compare SENs, hypercubes, and crossbars. A particular example, the Connection Machine, was also examined to obtain a real-world reference. It was clear that as larger levels of parallelism are employed, existing electrical connection technology would be hard pressed to provide the high degree of connectivity and parallelism necessary for high performance. Our results revealed that if a large number of boards are used in implementing the architecture, then a single-stage SEN is the best choice, provided that the network load is not very high.

Electrical interconnections were found to be inadequate in meeting the interconnection requirements of many parallel architectures. We therefore examined the properties of both demonstrated and emerging optical interconnection technologies. Rather than merely examine the performance of the interconnection medium, we considered the entire interconnection problem, including the possible implementation of the optoelectronic transducers required to interface with the electronic processing elements. Specific approaches investigated were fibers, polymer waveguides, planar holograms, volume holograms, and bulk optics/microoptics. Our assessment reveals that polymer waveguides offer the most promise if electronic processing elements are to be used in conventional architectures. The choice was driven significantly by the absence of suitable transducers to operate with three-dimensional free-space interconnects, rather than by predictions of attainable interconnection density based on diffraction limits.

A critical element in parallel optical interconnection is the optical source. For systems employing parallel operation of 1024 or more processors, considerations of lifetimes suggest that existing diode lasers will not provide adequate performance. Our favored approach involves employing a small number of high-power lasers in remote locations where their operation may be better controlled. These lasers are then fanned out to high-density modulator arrays. To be immune to the variations of temperature and wavelength likely to encountered in practical; machines, devices relying on the electrooptic effect offer the most promise.

A novel waveguide modulator was developed to meet the unique requirements of high density array uses. The device relies on polarization rotation and subsequent differential modal attenuation. This enables the unwanted light to be converted to heat. Problems associated with previously reported versions have been overcome by our novel design. Extinction ratios of up to 17dB have been determined for this device.

Arrays of electrooptic waveguide modulators were developed. Both the novel polarization rotator and the more conventional Mach-Zehnder modulators were used to construct arrays with interdevice spacings as small as 20 microns. Arrays of eight working devices were demonstrated. The Mach-Zehnder devices used had the highest extinction ratio reported to date for any III-V modulator, 23dB. The bandwidths of both devices were determined and found to be at least 1GHz. Higher frequency operation may be possible, but could not be verified using the test arrangement.

The optical components and the concepts developed here will enable optical interconnections to implement fully parallel connections within massively parallel, multi-board systems. While several key problems remain unsolved, particularly the need for custom receivers, parallel optical interconnects certainly appear to be a feasible solution to a serious impediment to parallel system design.

16. References

[1] D. A. Waterman, A Guide to Expert Systems, Addison-Wesley, 1985.

[2] F. Hayes-Roth, D. Waterman, and D. Lenat, Expert Systems, Addison-Wesley, 1983.

[3] J. Neff, "Optics and symbolic computing", Proc. SPIE **634**,24 (1987).

[4] C. Warde and J. Kottas, "Hybrid optical inference machines: architectural considerations", Appl. Opt. **25**, (15 March 1986) 940.

[5] R.A. Schmidt, W.T. Cathey, "Optical representations for artificial intelligence problems", Proc. SPIE **625**, (1986) 226.

[6] R.A. Schmidt and W.T. Cathey, "Optical representations for symbolic logic", Opt. Eng. 27(6), 475 (1988).

[7] A. McAulay, "Real-time optical expert systems", Appl. Opt. **26**, (May 1987) 1927; A. D. McAulay, "Optical Prolog computer using symbolic substitution", Proc. SPIE 881, 223 (1988).

[8] H.J. Caulfield, "Optical inference machines", Opt. Commun. **55**, 259 (15 Sept 1985).

[9] G. Eichmann and H.J. Caulfield, "Optical learning (inference) machines", Appl. Opt. **24**, 2051 (15 July 1985).

[10] D.P. Casasent and E.C. Botha, "Knowledge in optical symbolic pattern recognition processors", Opt. Eng. 26, 034(1987).

[11] H. Bartelt, S.K. Case, "Coordinate transformations via multifacet hologram optical elements", Opt. Eng. **22**, 497 (1983).

[12] A. Huang, "Impact of new technological advances and architectural insights on the design of optical computers", Proc. SPIE, **456**, (1984) 25.

[13] A. Huang, "Architectural considerations involved in the design of an optical digital computer", Proc. IEEE **72**, 780 (1984).

[14] B.K. Jenkins, P. Chavel, R. Forcheimer, A.A. Sawchuk, T.C. Strand, "Architectural implications of an optical computer", App. Opt. **23**, 3465 (1984).

[15] D. Psaltis and J. Hong, "Storage capacity of holographic associative memories", Opt. Lett. **11**, 812 (1986).

[16] D. Psaltis, Proceedings of the 34th Scottish University Summer School in Physics: Optical Computing, 14-26 August 1988, Heriot-Watt University, Edinburgh, U.K.

[17] M.E. Prise, N. Streibl, and M.M. Downs, "Optical considerations in the design of digital optical computers", Opt. and Quant. Electron. **20**, 49 (1988).

[18] S. Gregory, Design, Application and Implementation of a Parallel Logic Programming Language, Doctoral Dissertation, Department of Computing, Imperial College of Science and Technology, (September, 1985).

[19] E.Y. Shapiro, "Systems Programming in Concurrent Prolog", Proceedings of the 11th ACM Symposium of on Principles of Programming Languages, (January 1984).

[20] D. Turner, "A New Implementation Technique for Applicative Languages", Software-Practice and Experience, Vol. **9**, (1979).

[21] D. Psaltis and R.A. Athale, "High accuracy computation with linear analog optical systems: a critical study", Appl. Opt. **25**, 3071 (15 Sept. 1986).

[22] Arvind and V. Kathail, "A Multiple Processor Data Flow Machine that supports Generalized Procedures", Proceedings of the 8th Annual International Symposium on Computer Architecture, (May 1981).

[23] R. Ramnarayan, et al, "Very Large Parallel Data Flow", RADC Final Technical Report, RADC-TR-88-42, Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, NY (March 1988).

[24] B.W. Arden and H. Lee, "Analysis of chordal ring network",IEEE Transaction on Computers, Vol. C-30, 1981, pp. 291-295.

[25] W.F. Clocksin, and C.S. Mellish, Programming in Prolog, Springer-Verlag, New York, (1981).

[26] A. Huang, "Optical digital computers?" Digest of Papers COMPCON Spring 1985. Thirtieth IEEE Computer Society International Conference (25-28 Feb 1985), 290.

[27] Y. J. Hsu, et al, "A Gate-Array Design of a Shuffle-Exchange Network Switching Element", 6th Biennial University/Government/Industry Microelectronics Symposium, 1985.

[28] R. Kostuk, Doctoral Dissertation, Stanford University, 1986.

[29] M. Murdocca and N. Streibl, "A Digital Design Technique for Optical Computing", Proc. of the Topical Meeting on Optical Computing (1987), Lake Tahoe, NV, Paper MB2.

[30] D.H. Hartman, "Digital high speed interconnects: a study of the optical alternative", Optical Engineering, October 1986, Vol. 25, No. 10, pp. 1086 - 1102.

[31] K.H. Böhle, Lectures given at 34th Scottish University Summer School in Physics, 14-26 August 1988, Heriot-Watt University, Edinburgh, Scotland.

[32] K. Wagner and D. Psaltis, "Multilayer optical learning networks", Appl. Opt. 26, 5061(1988).

[33] H. Rajbenbach, Proceedings of the 34th Scottish University Summer School in Physics: Optical Computing, 14-26 August 1988, Heriot-Watt University, Edinburgh, Scotland, to be published.

[34] A.A. Sawchuk, and T.C. Strand, "Digital Optical Computing", Proc. IEEE **72**, (1984) 758.

[35] S.D. Smith, A.C. Walker, B.S. Wherrett, F.A.P. Tooley, J.G.H. Matthew, M.R. Taghizadeh, and I. Janossy, "Cascadeable digital optical logic circuit elements in the visible and infrared: Demonstration of some first all-optical circuits", App. Opt. **25**, 1586 (1986).

[36] M. Derstine and A. Guha, "Design Considerations for an Optical Symbolic Processing Architecture",Optical Engineering, Special Issue on Optical Computing, April 1988.

[37] D. Turner, "Combinator Reduction Machines", Proceedings of the International Workshop on High-Level Computer Architectures, University of Maryland, May 1984.

[38] R. Wilensky, LISPcraft, W.W. Norton & Company, New York, 1984.

[39] T. Kushner, et al, "Image Processing on the ZMOB", IEEE Transactions on Computers, October 1982, pp. 943 - 951.

[40] B. Wherrett, "Optical Computing", Proceedings of the 4th European Conference on Integrated Optics, Glasgow, Scotland, May 11-13, 1987.

[41] K.-H. Brenner, A. Huang, N. Streibl, "Digital optical computing with symbolic substitution", Appl. Opt. **25**, (15 Sept. 1986), 8054.

[42] T.J.W. Clark, et al, "SKIM - the S, K, I Reduction Machine", Proceedings of the 1980 ACM LISP Conference, 1980, pp. 128 - 135.

[43] M. Scheevel, "NORMA: A Graph Reduction Processor", 1986 ACM Conference on LISP and Functional Programming Languages, August 1986.

[44] A.W. Lohmann, "What Classical Optics can do for the Digital Optical Computer", and "Optical Perfect Shuffle", Applied Optics, 15 May 1986, Vol. 25, No. 10.

[45] W.D. Hillis, "The Connection Machine", MIT Press, 1985.

[46] D.H. Lawrie and D.A. Padua, "Analysis of Message Switching with Shuffle- Exchanges in Multiprocessors", Proc. of the Workshop on Interconnection Networks for Parallel and Distributed Processing, 1980, pp. 116 - 123.

[47] S. Abraham and K. Padmanabhan, "Performance of the Direct Binary n-Cube Network for Multiprocessors", Proc. International Conference on Parallel Processing, 1986, pp. 636 - 639.

[48] T. Lang, "Interconnections Between Processors and Memory Modules Using the Shuffle-Exchange Network", IEEE Transactions on Computers, May 1976, pp. 496 - 503.

[49] D.H. Lawrie, "Access and Alignment of Data in an Array Processor", IEEE Transactions on Computers, December 1975, pp. 1145 - 1155.

[50] D.M. Dias and J.R. Jump, "Analysis and Simulation of Buffered Delta Networks", IEEE Transactions on Computers, April 1981, pp. 273 - 282,

[51] J.H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors", IEEE Transactions on Computers, October 1981, pp. 771 - 780.

[52] W.D. Hillis, "The Connection Machine", MIT Press, 1985.

[53] J.E. Midwinter, "Novel Approach to the Design of Optically Activated Wideband Switching Matrices", IEE Proceedings, Vol. 134, Pt. J, No. 5, October 1987; "'Light' Electronics, myth or reality", IEE Proceedings, Vol. 132, Pt. J, No. 6, December 1985.

[54] G. Eichmann and Yao Li, "Compact Optical Generalized Perfect Shuffle", Applied Optics, 1 April 1987, Vol. 26, No. 7.

[55] R.Jin et al. "All Optical Compare and Exchange Switches", I.E.E.E. Selected Areas in Communications, Vol 6, No 7, 1988, pp 1273-1279.

[56] J. Shamir, H. J. Caulfield, W. Micelli, and R. J. Seymour, "Optical Computing and the Fredkin Gates", Applied Optics, 15 May 1986.

[57] R. Cuykendall and D. McMillin, "Control-specific Optical Fredkin Circuits", Applied Optics, 15 May 1987

[58] E. Fredkin and T. Toffoli, "Conservative Logic", International Journal of Theoretical Physics, Vol. 21, Nos 3/4, 1982, pp. 219 - 253.

[59] Clark, N. A., M. R. Meadows, M. A. Handschy and K. M. Johnson, "Optical Interconnections Using Ferroelectric Liquid Crystals", OSA Annual Meeting, Technical Digest, Rochester, NY, p. 119, Oct. 1987.

[60] McManus, J. B., R. S. Putnam and H. J. Caulfield, "Demonstration of Switched Holograms for Optical Interconnection", OSA Annual Meeting, Technical Digest, Rochester, NY, p. 120, Oct. 1987.

[61] A. Guha, R. Ramnarayan, and M. Derstine, "Architectural Issues in Designing Symbolic Processors in Optics", Proceedings of the 14th Annual International Symposium on Computer Architecture, (June 1987).

[62] Effron, U, "Spatial Light Modulators and their Applications", Proc SPIE, Vol 700, 1986, pp132-145

[63] AT&T, commercial literature for multifiber array connector (MAC), Spring 1988.

[64] R. Selvaraj, et al, "Integrated Optical Waveguides in Polyimide for Wafer Scale Integration", Journal of Lightwave Technology, Vol. 6, No. 6, July 1988.

[65] "General Aspects of the Selfoc Lens Technology", and other literature available from the Nippon Sheet Glass Co. through NSG America, Inc., 28 World's Fair Drive, Somerset, NJ 08873.

[66] J.D. Zook, T.C. Lee, "Geometrical Interpretation of Gaussian Beam Optics", Applied Optics, Vol. 11, page 2140, 1972.

[67] M. Izutsu et al. "Operation Mechanism of the Single-Mode Optical Waveguide Y-Junction", Applied Optics, Vol. 21, page 136, 1982.

[68] U. Niggebrugge et al. "Self-Aligned Low-Loss Totally Reflecting Waveguide Mirrors in InGaAsP/InP", Proceedings of the Fourth European Conference on Integrated Optics, C.Wilkinson and J. Lamb editors, Glasgow, U.K, May 11-13, 1987, pp. 90-93.

[69] K.W. Murphy "An Integrated Optics Technology for the Production of Photocor Fiber-Optic Components", Corning Glass Works Technical Report, 1987.

[70] J. Mendoza Alvarez et al. "Analysis of Depletion Edge Translation Lightwave Modulators", I.E.E.E. Journal of Lightwave Technology, Vol LT-6, No. 6, 1988, pp793-808

[71] M. Belanger et al. "Fabrication and Characterization of a Linear Mode Confinement Modulator on GaAs", I.E.E.E. Journal on Selected Areas in Communications, Vol. 6, No. 7, August 1988, pp. 1205-1208.

[72] F. Rottmann et al. "Integrated-Optic Wavelength Multiplexers on Lithium Niobate Based on Two-Mode Interference", I.E.E.E. Journal of Lightwave Technology, Vol. 6, No. 6, June 1988, pp. 946-952.

[73] L. Thylen, "Integrated Optics in Lithium Niobate: Recent Developments in Devices for Telecommunications", ibid, pp. 847-861.

[74] See for example: A. Otto, "Excitation of Nonradiative Surface Plasma Waves in Silver by the Method of Frustrated Total Reflection", Zeitschrift fur Physik, Vol. 216, pp. 398-410, 1968.

[75] M. S. Stern, "Semivectorial Polarized Finite Difference Method for Optical Waveguides with Arbitrary Index Profiles", I.E.E. Proceedings Part J, Vol. 135, No. 1, February 1988, pp. 56-63.

[76] L.M Johnson and F.J.Leonberger "Low Loss $LiNbO_3$ Waveguide Bends with Coherent Coupling"

[77] R. Walker and A. Carter, presented at the S.P.I.E. Conference on Integrated Optical Circuit Engineering, Boston, September 1988.

[78] S. Wang and H. Lin "High-Speed III-V Electro-optic Waveguide Modulators at 1=1.3 I.E.E.E. Journal of Lightwave Technology, Vol. 6, No. 6, June 1988, pp. 758-771.

[79] See for example: E. J. Murphy, "Fiber Attachment for Guided Wave Devices", I.E.E.E. Journal of Lightwave Technology, Vol. 6, No. 6, June 1988, pp. 862-871.

[80] C. Harder, et al, "5.2 GHz Bandwidth Monolithic GaAs Optoelectronic Receiver", IEEE Electron Devices Letters, Vol. 9, No. 8, April 1988.